Second Annual

OS/2 World Conference & Exhibition

Proceedings — Volume 2

July 17-20, 1995
Hynes Convention Center
Boston, Mass.

Miller Freeman
A United News & Media company

this page intentionally left blank

Second Annual

OS/2 World Conference & Exhibition

Proceedings — Volume 2

ISBN #0-87930-397-2

©Copyright 1993 by Miller Freeman, Inc., and as designated. All rights reserved. No part of this book may be reproduced or copied in any manner whatsoever without the express written consent of Miller Freeman, Inc.

Miller Freeman, Inc. • 600 Harrison Street • San Francisco, CA 94107 • (415) 905-2200

this page intentionally left blank

Table of Contents

LAN 0709 Increasing Server Performance with Full Duplex LAN Switching 9
Laura Jeanne Knapp

MGT 0702 Setting Up an Internal Support Organization 35
David Moskowitz

MGT 0705 15 Steps to Downsizing to OS/2 41
Robert Chapman

MM 0708 Multimedia Creation and Management 57
Peter Yanker

PRO 0703 Writing Reusable Classes with C++ 81
Noel Bergman

PRO 0706 IBM COBOL goes Desktop! 89
Ron Langer

PRO 0707 Migrating from Graphical Applications to Object-Oriented User Interfaces 101
Theo Mandel

SYS 0701 Configuring Windows Under OS/2 121
Mark Minasi

SYS 0710 Using OS/2 to Reach the Internet 127
David Gaertner

MGT 0809 The OS/2 Problem Solver 137
David Moskowitz

PRO 0801 Using Visual REXX to Produce Graphical Utilities 145
Bill Schindler

*No paper available; contact speaker for further information

PRO 0802 Lotus Notes as a Development Tool (Using REXX) 151
Brian Proffit

PRO 0803 Programming with Distributed Objects 161
Roger Sessions

PRO 0805 SOM Metaclass Programming 175
Ira Forman

PRO 0807 C Set ++ Class Library 193
Pete Devenyi

SYS 0804 Mastering NT Server 205
Mark Minasi

SYS 0806 Building Successful DB2/2 Applications 211
Mike Logan

SYS 0808 Network Security and Single Signon 225
Barbara Porebski

CS 0901 OS/2 Database Design and Programming 243
Rob Veitch

CS 0902 DB2 Family: DB2 Technology at Work 249
Dan Britton

CS 0903 Introduction to APPC and APPN 275
Dave Lutz

LAN 0904 What's New in IBM TCP/IP for OS/2? 293
John Conner

LAN 0906 Configuring Multiple Protocols in a Workstation 297
Laura Jeanne Knapp

*No paper available; contact speaker for further information

MGT 0911 How to Introduce Objects to Your Organization 313
David Moskowitz

PRO 0905 The Power of Frameworks 321
David Sharp

PRO 0907 VisualAge—A Powerful New Vision of Programming * 339
Alan Friedman

SYS 0908 Compatibility—Today's Buzzword That Makes a Difference 341
Joyce Batty

SYS 0909 Education and the Evolution of OS/2 * 357
Feite Kraay

CS 1009 Advanced Database Recovery with DB2/2 359
Chris Fierros

LAN 1010 LAN Automated Distribution/2 377
Bryan Heft

MNC 1004 Communications Manager/2: Advanced Configurations 381
Don Richards

PRO 1003 The OS/2 Workplace Shell (Accept no substitutes) * 417
Evan Bergman

PRO 1005 Object Persistence: Beyond Object-Oriented Databases 419
Roger Sessions

PRO 1006 Making Concurrent Messages under OS/2 2.X 431
Luca Miccoli

PRO 1008 Visual Builder for C Set ++ 453
Dale Nilsson

*No paper available; contact speaker for further information

SYS 1002 Getting It Right: Testing OS/2 Applications 463
Phil Arthur

SYS 1007 Migrating Mainframe Programmers to OS/2 469
John Montgomery

CS 1106 DB2/2 Database Administration 477
Rick Weaver

MNC 1108 Creating a CID Enabled Installation Using Software Installer 489
Ken Greenlee

PRO 1105 Understanding and Exploiting OS/2 Grapical Programming 505
Kelvin Lawrence

PRO 1107 Using the APPC Application Suite and Discovery 533
Dave Lutz

SYS 1101 Building Real-World Computer Benchmarks 547
Robert Chapman

SYS 1102 Keep 'em Running: How to Support End-Users 565
Esther Schindler

SYS 1104 Implementing the GSS-API in your Client/Server Application 573
Barbara Porebski

SYS 1110 32-bit Migration Tips and Techniques * 581
Daniel Kardell

CS 1209 Build an OS/2 Client/Server Application * 583
Bill Rooks

*No paper available; contact speaker for further information

LAN 1202 Optimizing the Netware Client for OS/2 585
Bob Panza

MM 1210 Producing Your Own CD-ROMs 595
Guy Wright

MNC 1206 Integrating NetWare Into the OS/2 Environment * 609
Steve Baumgarner

MNC 1207 Performance Tuning OS/2 611
Ron Cadima

MNC 1208 Technical Coordinator Program 625
Bob Gress

PRO 1203 What's Wrong with Object-Oriented Programming 633
Bill Schindler

PRO 1204 Agency: Mobile Agents in Object REXX 637
Rick McGuire

SYS 1205 OS/2 Systems Management Toolbox 643
Craig Elliott

*No paper available; contact speaker for further information

this page intentionally left blank

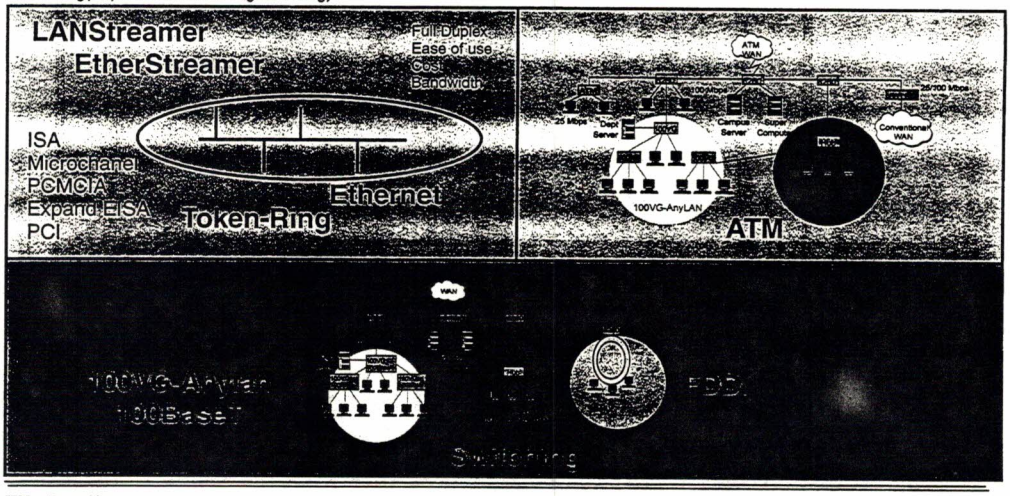
LAN 0709
Increasing Server Performance with Full Duplex LAN Switching

Laura Jean Knapp
IBM

IBM Networking

Connecting people to information through technology

IBM LAN Technology Support

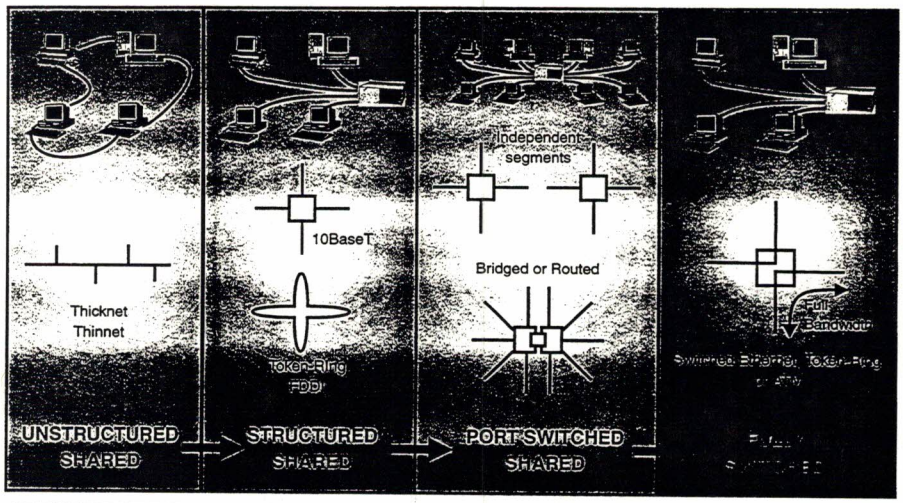


IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_001A

IBM Networking

Connecting people to information through technology

Networks in Transition

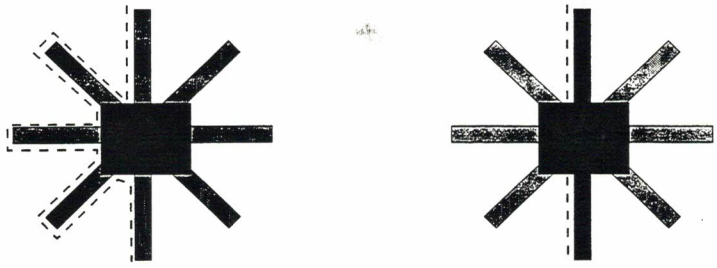


IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_002A

IBM Networking

Connecting people to information through technology

Shared Media versus Switched LANs



Shared Media

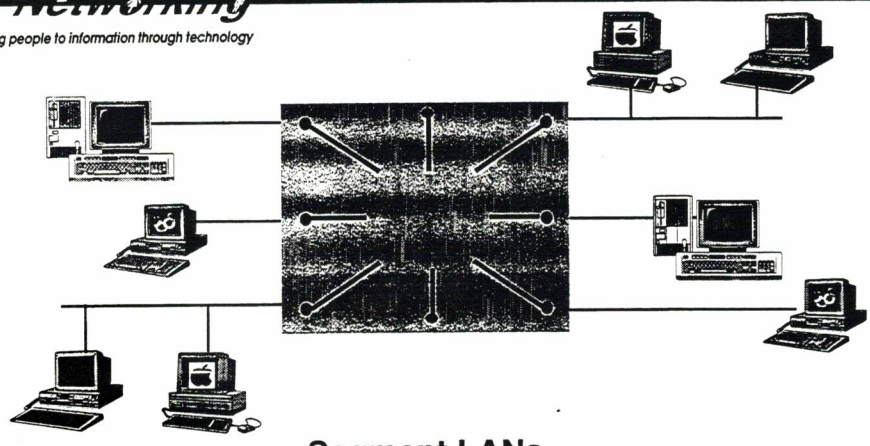
Switched

© Copyright IBM Corporation 1993, 1994

IBM Networking

Connecting people to information through technology

Switched LANs



Segment LANs

Attach dedicated port or shared LAN

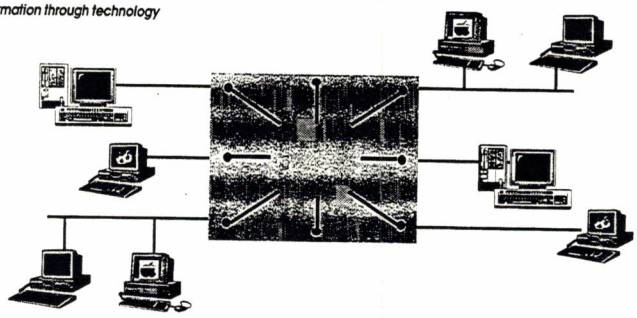
Low latency packet switch

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Switched LANs - Usage



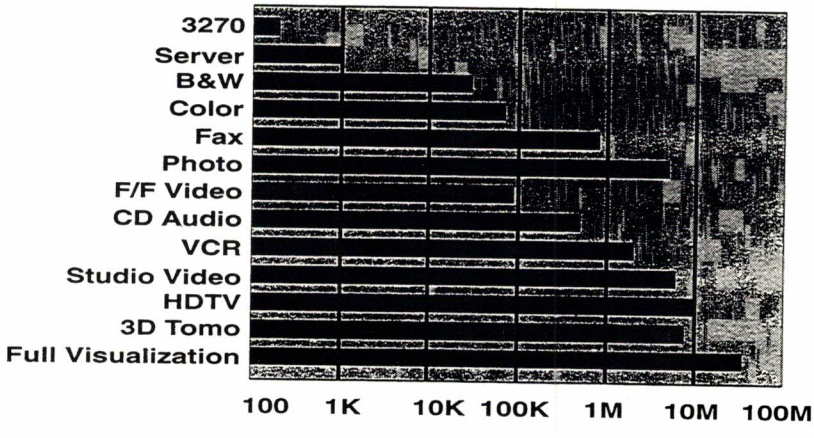
Increases bandwidth
Parallel switching
Very low latency
Appropriate for small number of connections
Effective when traffic distributed over multiple sessions

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Why do we need Switching LANs ?



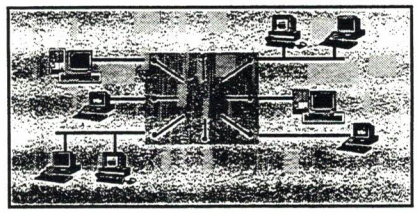
© Copyright IBM Corporation 1994, 1995

IBM Networking

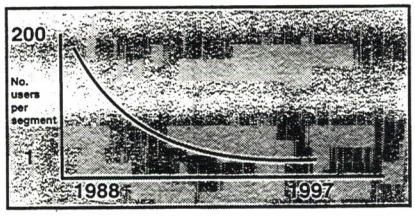
Connecting people to information through technology

Switching - Ultimate LAN Segmentation

Faster devices



Constant network redesign



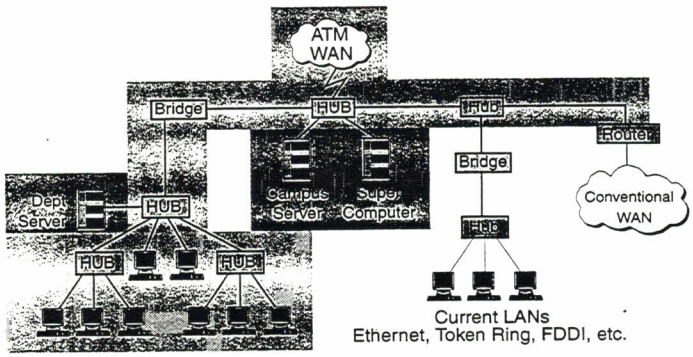
User expectations

© Copyright IBM Corporation 1994, 1995

IBM Networking

Connecting people to information through technology

Where Are Your Network Problems?



- Workgroups with demanding applications like video-conferencing, interactive imaging, multimedia
- Connections to ATM networks outside the campus
- Campus backbone combining heavy traffic, multiple media, and isochronous req'ts
- Critical resources with many requestors, high bandwidth req'ts, and multimedia

© Copyright IBM Corporation 1994, 1995

IBM Networking

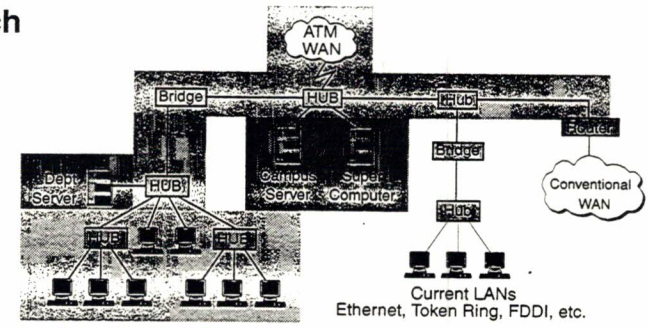
Connecting people to information through technology

Switched or Higher Speed ?

Server technology is a key factor

Switch

386
486



High Speed

Pentium
RISC
Mini
Mainframe

IBM Networking © Copyright IBM Corporation 1994, 1995 FDX_006D

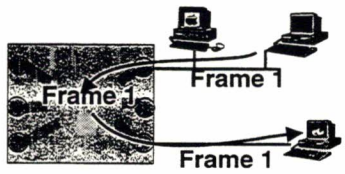
IBM Networking

Connecting people to information through technology

Cut-Through vs Store-Forward Switches

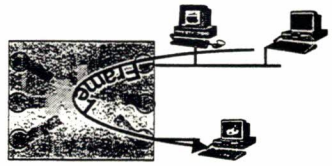
Store and Forward Operation

- Entire packet received prior to retransmission
- FCS verified prior to forwarding
- Short packets queued behind long packets
- Used by Bridges today
- Supports different speed LAN switching
- 800 ms latency



Cut Through Operation

- Read header prior to retransmission
- No storage reduces latency
- FCS not verified prior to forwarding
- Revert to store-forward at burst loads
- Same speed LANs only
- 50ms latency



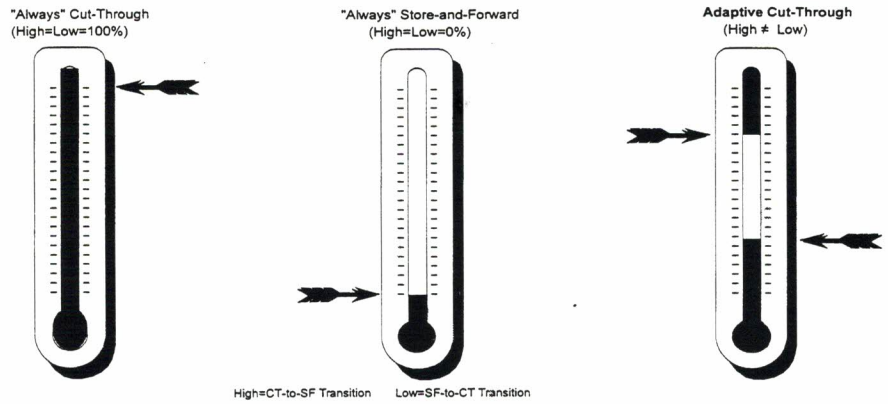
IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_006E

Adaptive Cut-Through Switching

IBM Networking

Connecting people to information through technology

User-Specified Error Rate Thresholds



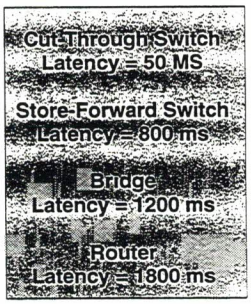
© Copyright IBM Corporation 1993, 1994, 1995

Switches, Bridges, and Routers

IBM Networking

Connecting people to information through technology

- Switches
  - Simpler Functions
  - Optimized for Local Homogeneous Connections
  - Optimized for Layer 2 Filtering and Forwarding
  - Lower Price per Port
  - Better Performance (Lower Latency)
- Bridges and Routers:
  - More Complex Functions
  - Optimized for mixed LAN, WAN Connections
  - Optimized for Layer 2&3 Filtering and Forwarding
  - Higher Price per Port
  - Lower Performance (Higher Latency)



© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

What is Full Duplex Switching?

Simplex

One-way transmission
Public Address System

Half-Duplex

Directional two-way transmission
Speakerphones
Standard Token-Ring and Ethernet

Full-Duplex

Simultaneous, two-way transmission
Telephone System
Full-Duplex Token-Ring and Ethernet

© Copyright IBM Corporation 1994

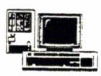
IBM Networking

Connecting people to information through technology

Full-Duplex Switched Ethernet Today !

IBM 8271 Etherstreamer Switch

Full-duplex transmission
Media speed transfers
Low latency (less than 50 microseconds)
8 shared or dedicated 10BaseT ports
SNMP management
Cascading
EtherStreamer adapter for full duplex support
All Ethernet adapters and repeater support in switch mode

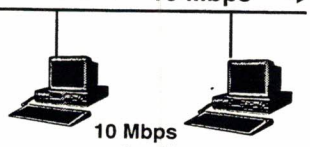


20Mbps

10 Mbps

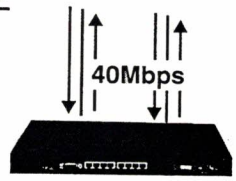
Model 108

Universal features slot (UFS)
EtherProbe Port
Adaptic Cut-Through Support



10 Mbps
one direction
at a time

40Mbps

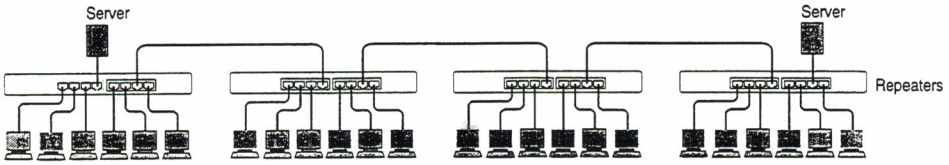


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Typical Ethernet LAN



Current Congested Ethernet LAN Example

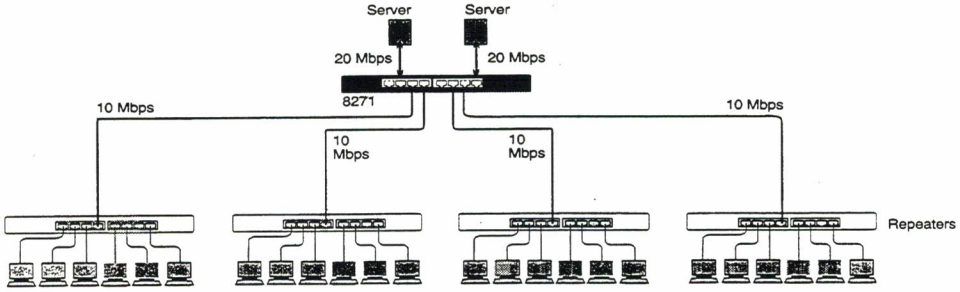
- **24 Clients and 2 Servers**
- **Single Collision Domain**
- **All Stations (Clients and Servers) Share 10 Mbps**

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_008B

IBM Networking

Connecting people to information through technology

8271 Simple Microsegmentation



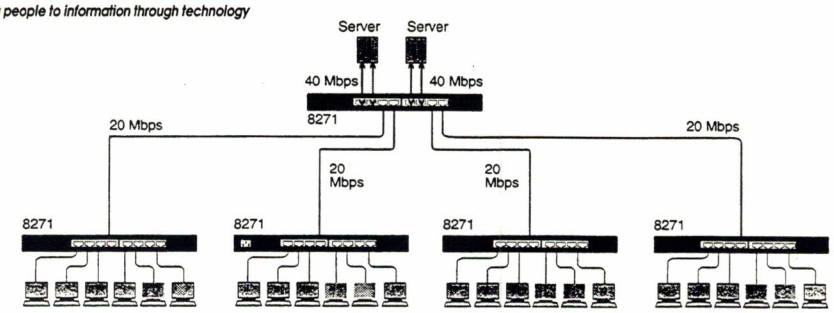
Separate 24 clients into 4 segments
Each server on dedicated segment

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_008C

8271 Migration to Dedicated LAN Segments

IBM Networking

Connecting people to information through technology



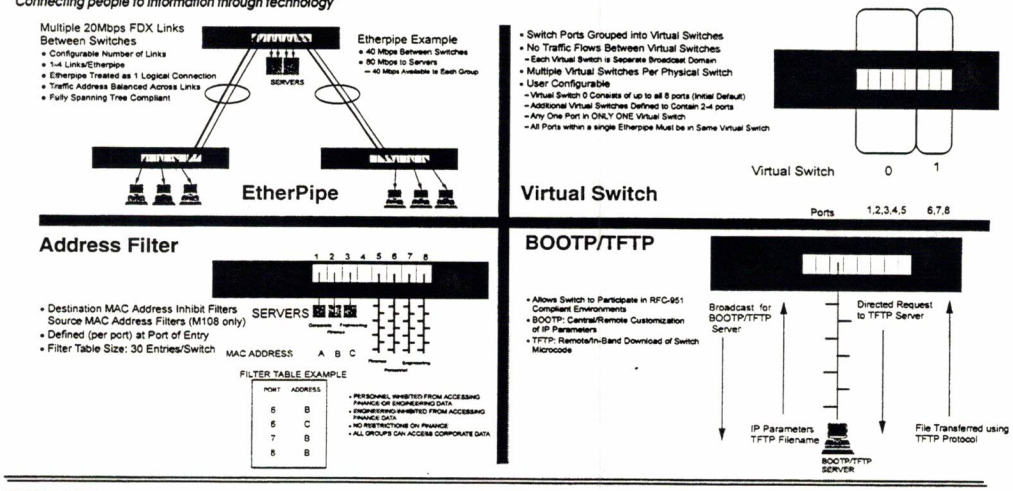
- All Stations Each on Dedicated Segment
  - 10 Mbps Bandwidth to Every Client
  - 20 Mbps Bandwidth to Clients w/ Full Duplex Adapters
- Servers on Dedicated Full Duplex Segments
  - Double Server Bandwidth Using Dual Ports
- Up to 8x Improvement in Total Network Bandwidth

© Copyright IBM Corporation 1993, 1994, 1995

8271 Advanced Functions

IBM Networking

Connecting people to information through technology



© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8271 Futures

One-port ATM (155Mbps SONET) -- {1H96}
- Multimode fiber connection to ATM switch
- Forum compliant LAN Emulation

One-Port 100BaseT -- {4Q95}
- Connects to 10BaseT repeater

Multiport 100VG AnyLAN (Ethernet) -- {4Q95}
- Connects to 100VG Concentrator

Multiport 10BaseT --{4Q95}
- Expands 8271 Model 108 to more than 8 ports

Multiport 10BaseFL -- {4Q95}
- Fiber connections to remote ethernet repeaters

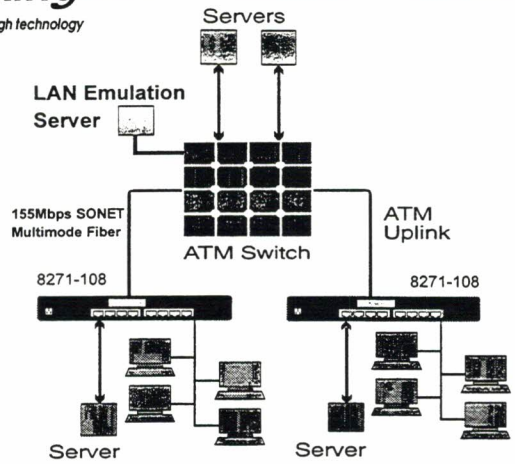
Feature of the 8260 -- {1H96}

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8271 Integration with ATM



*Preview. Tgt Availability: 1H96

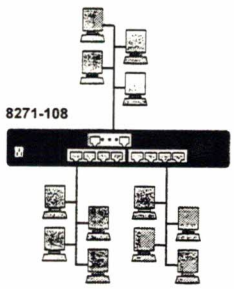
© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

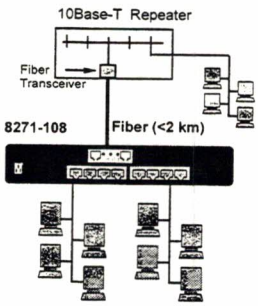
8271 Connectivity Options

10Base-T UFC*



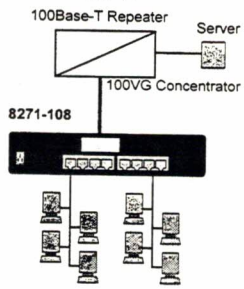
Expands 10Base-T Port Density

10Base-FL UFC*



Connects Remote 10 Mbps Ethernet Segments

100Base-T or 100VG UFC*



Switch Ethernet Segments to High Speed Servers

*Product Preview. Target Availability: 4Q95

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8271 Management

- Network Management (Configuration/Operational Control)
  - Console Interface Accessible via–
    - VT-100 (or equivalent) Connected to Serial Port (Directly connected or via serial modem)
    - TELNET
  - Management Functions Also Accessible through SNMP Agent
    - Management Information Base (MIB) Definition Shipped with Switch
    - Manageable from ANY SNMP Management Station (eg. NetView for AIX, HP OpenView)
- 8271 NetView for Windows program (separate program)
  - Based on IBM NetView for Windows platform
    - Low-Cost SNMP Network Management Platform
    - MS Windows Environments
  - Other IBM Networking Products with NetView for Windows Support
    - IBM 8224 Ethernet Stackable Hub
    - IBM 8230 Model 3 Controlled Access Unit

RFC Number	Description
1213	MIB-II
951	Bootstrap Protocol (BOOTP)
1493	Bridge MIB
1573*	Interfaces Extension MIB
1643*	Ethernet Interface MIB

- 8271 NetView for AIX Program (separate program)
  - Based on IBM NetView for AIX platform
    - Full-Function SNMP Management Platform
    - AIX Environments
    - Full Suite of Related Management Applications

© Copyright IBM Corporation 1993, 1994, 1995

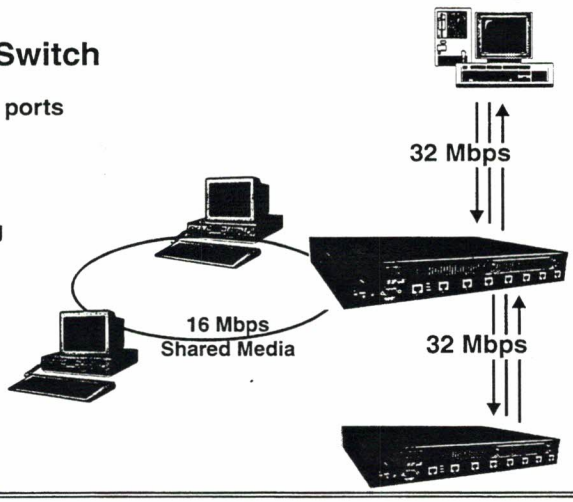
IBM Networking

Connecting people to information through technology

8272 LANStreamer Switch

IBM 8272 Token-Ring Switch

- 8 shared or dedicated 10BaseT ports
- Full-duplex transmission
- Media speed transfers
- Adaptive cut-through switching
- Cut-through switching
- Sotre-forward swtiching
- STP and UTP support
- Universal Feature Slot
- SNMP management

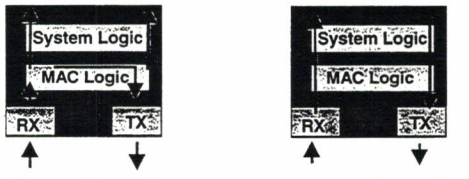


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Full-Duplex Switched Token-Ring



Suspend unused MAC Functions
- Active/passive monitor
- Ring recovery functions
- All token access control mechanisms

Full Duplex transmit/receive
- Transmit without token-wait
- Receive all incoming packets

Token-less
Ring-less
Token-Ring

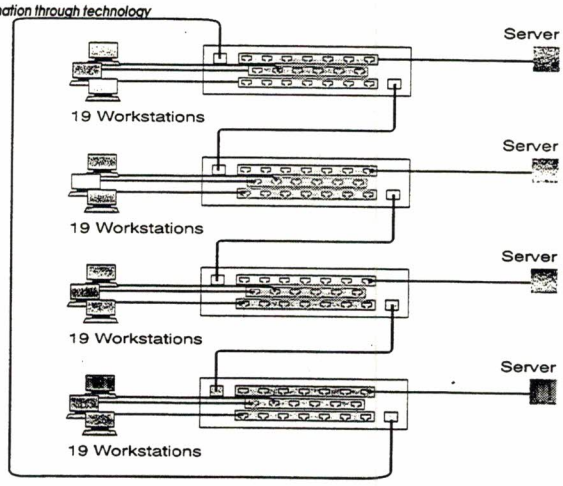
Physical level clocking and timing
- Stations transmit on local crystal
- Stations receive on derived clock

© Copyright IBM Corporation 1993, 1994, 1995

Typical Token-Ring Configuration

IBM Networking

Connecting people to information through technology

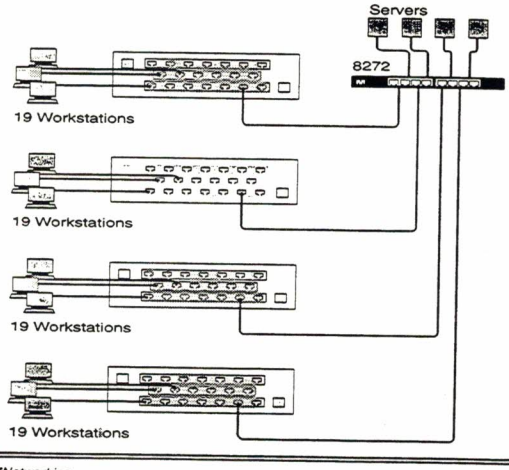


© Copyright IBM Corporation 1993, 1994, 1995

8272 Microsegmentation

IBM Networking

Connecting people to information through technology



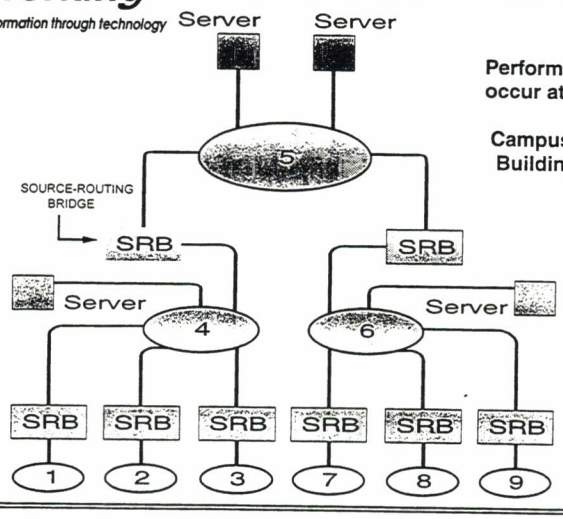
- Divide76 Clients into 4 Segments; 19 Clients Share 16 Mbps
- Each Server on Own Segment
  - Each Has Dedicated 16 Mbps Bandwidth
- Benefits
  - 4X Total Network Bandwidth (64 Mbps)
  - No Change to Client or Server Adapters
  - No Change to Wiring
  - No Change to Software
- Upgrading Server Adapters to FDX will Provide Some Additional Latency Improvement

© Copyright IBM Corporation 1993, 1994, 1995

Typical Multisegment Token-Ring

IBM Networking

Connecting people to information through technology



Performance problems occur at:

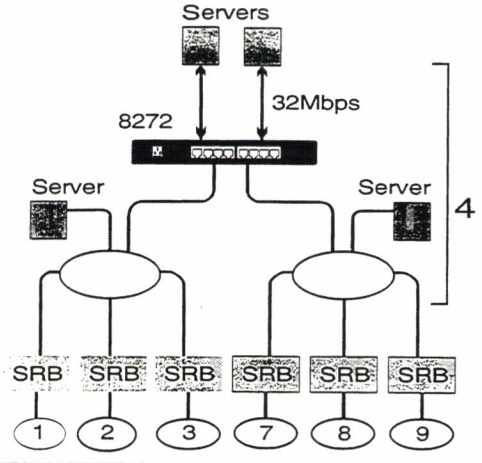
Campus Backbone (ring 5)
Building Backbone (ring 4/6)

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_009E

Backbone Congestion Solution with 8272

IBM Networking

Connecting people to information through technology



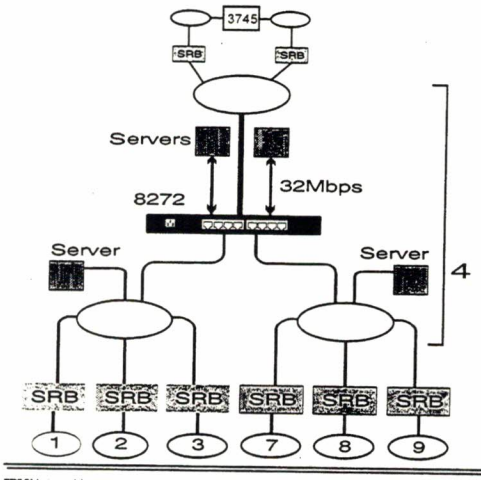
- Congested Backbone Ring (5) and attached SRBs replaced by 8272 LAN Streamer Switch
- Upgrade Campus Server Adapters to FDX
  - Servers Have Independent RW Paths
  - 4X Bandwidth to Campus Servers
- Bridges to 7/8/9 Reconfigured to Bridge to Ring 4, (was Ring 6)
- Additional Server Ports will Further Improve Latency

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_009F

8272 - Dual 3745 Token-Ring Connections

IBM Networking

Connecting people to information through technology



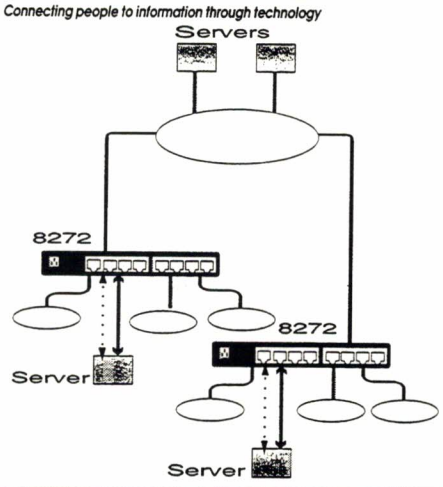
- Same Bandwidth Improvement for Campus Servers
- Host Connection Duality Maintained
- Host Connection No Longer Contends with Server Traffic
  - Full 16 Mbps Dedicated To Host Traffic
- SRBs for Host Connections Reconfigured to Bridge to Ring 4

© Copyright IBM Corporation 1993, 1994, 1995

8272 for a Congested Workgroup LAN

IBM Networking

Connecting people to information through technology



- Replace Workgroup Rings 4 & 6 (and the Bridges Around Them) with 8272
- Connect Workgroup Servers Directly to 8272
- All SRBs Eliminated;
  All Rings have Same Number (Topology Simplified, Hop Counts Eliminated)
- 2X Bandwidth to Workgroup Servers;
  Even More Improvement w/Addl Server Ports

© Copyright IBM Corporation 1993, 1994, 1995

8272 Ease of Use

IBM Networking

Connecting people to information through technology

- Adaptive Cut-Through Switching
  - User-Specified Thresholds
  - Self-Healing and Self-Optimizing
- Auto-Sense / Auto-Configure of Port Connection Type
  - Shared or Dedicated Media Segment
  - Half-Duplex or Full-Duplex
  - 4 Mbps or 16 Mbps
  - Connection to Other 8272
  - Current Token-Ring Lobe Cabling Support for ALL Types; NO Cross-Over Cables Required for Full Duplex or 8272-8272
- Source-Route Switching*
  - NO Ring/Bridge Number Configuration Required
  - Switch and all Directly Attached Stations/Segments Share Same Ring Number
  - Switch based on either MAC Address or Routing Information Field (RIF)
  - Supports Redundant Paths/Duplicate Addresses using External Bridges
  - IBM Leading Definition in IEEE 802.5 Standards Committee

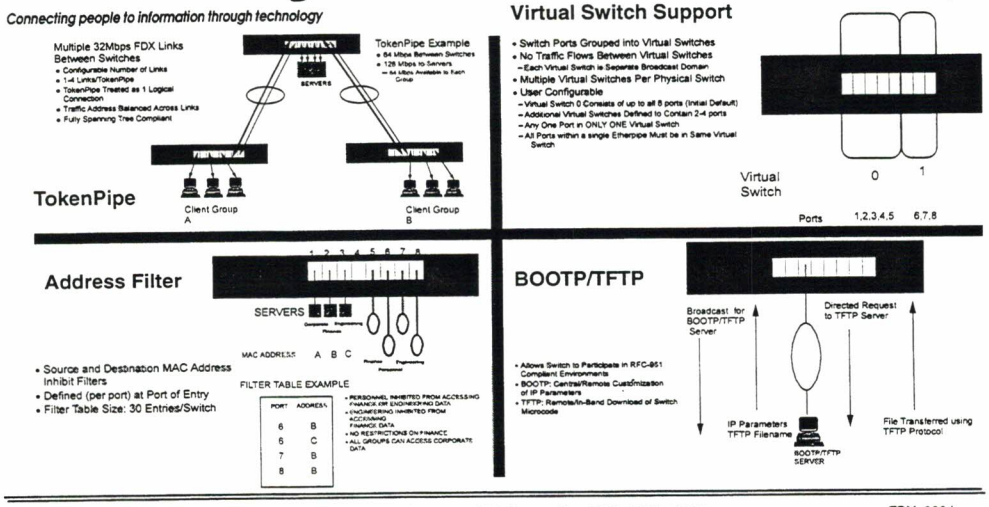
*Available to all 8272 customers at no additional charge via microcode upgrade by 4Q95

© Copyright IBM Corporation 1993, 1994, 1995

8272 Advanced Functions

IBM Networking

Connecting people to information through technology

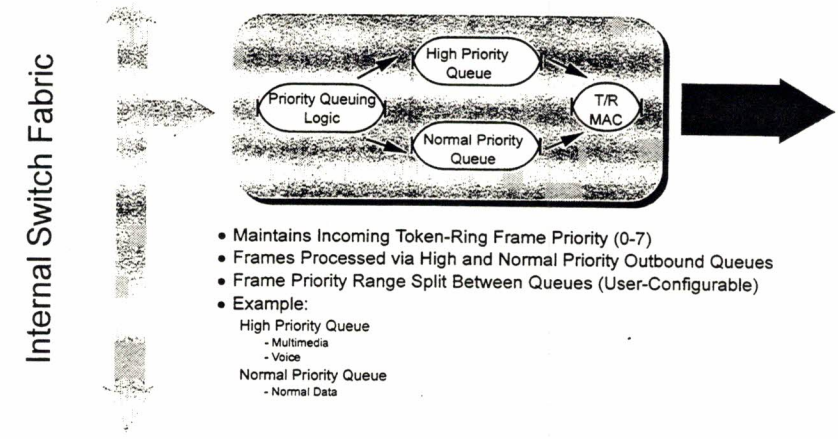


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8272 Advanced Functions



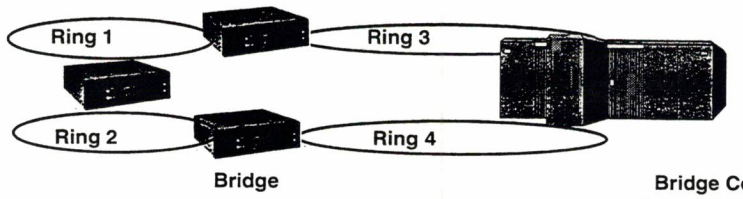
- Maintains Incoming Token-Ring Frame Priority (0-7)
- Frames Processed via High and Normal Priority Outbound Queues
- Frame Priority Range Split Between Queues (User-Configurable)
- Example:
  - High Priority Queue
    - Multimedia
    - Voice
  - Normal Priority Queue
    - Normal Data

© Copyright IBM Corporation 1993, 1994, 1995

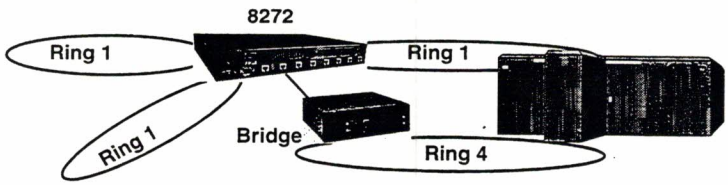
IBM Networking

Connecting people to information through technology

8272 Integration into Source-Routed Networks



Bridge Configuration



Switch Configuration

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8272 Management

- Network Management (Configuration/Operational Control)
  - Console Interface Accessible via--
    - VT-100 (or equivalent) Connected to Serial Port (Directly connected or via serial modem)
    - TELNET
  - Management Functions Also Accessible through SNMP Agent
    - Management Information Base (MIB) Definition Shipped with Switch
    - Manageable from ANY SNMP Management Station (eg. NetView for AIX, HP OpenView)
- Graphical Network Management Application Programs
  - NetView for Windows Support(*)
  - NetView for AIX Support(*)

RFC Number	Description
1213	MIB-II
951	Bootstrap Protocol (BOOTP)
1493	Bridge MIB
1573	Interfaces Extension MIB
1231	Token-Ring MIB

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8272 Futures

One-port ATM (155Mbps SONET) -- {1H96}
Multimode fiber connection to ATM switch
Forum compliant LAN Emulation

Multiport 100VG AnyLAN (Token-Ringt) -- {4Q95}
Connects to 100VG Concentrator

Multiport Token-Ring UTP/STP -- {4Q95}
Expands 8272 Model 108 to more than 8 ports

Multiport Token-Ring Fiber -- {4Q95}
Fiber connections to remote Token-Rings

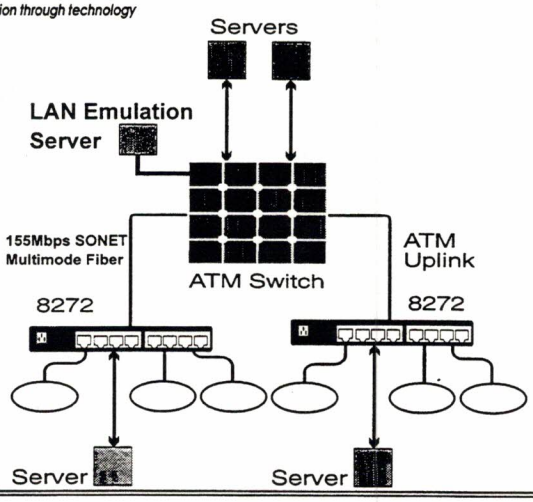
Feature of the 8260 -- {1H96}

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8272 Integration with ATM



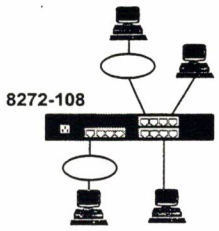
© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

8272 Connection Options

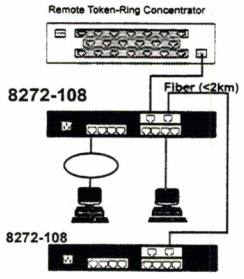
Token-Ring UTP/STP UFC*



- Additional 4/16Mbps UTP/STP Ports

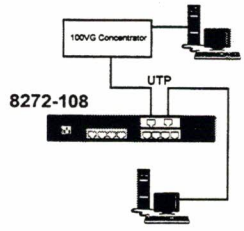
*Product Preview.
Target Availability: 4Q95

Token-Ring Fiber UFC*



- Switch Remote 4/16 Mbps Token-Ring Segments

100VG AnyLAN UFC*



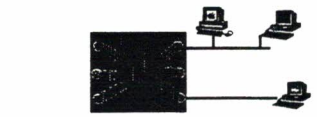
- Switch Token-Ring Segments to High Speed Servers

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Switched Issues

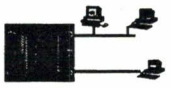


Processor Based
RISC processor
Management in data path
Limited by software efficiency

ASIC Based
Vendor specific
Management not in the data path
Performance limited by network bandwidth

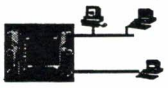
Internal Structure

Shared memory



All ports share common memory
Blocking architecture
Broadcast efficient

Distributed memory

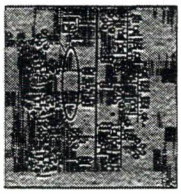


Dedicated buffer memory per port
Non-blocking architecture
Broadcast inefficient

Buffering critical to hold queued frames
Adds most cost to switched hub

Buffering

Bridge Integration



How many MAC address does switch hold?

IBM 8271 Etherstreamer Switch
6000 master table
1700 for each port

When port table is full the switch becomes a bridge and reads each frame

Can addresses be filtered?

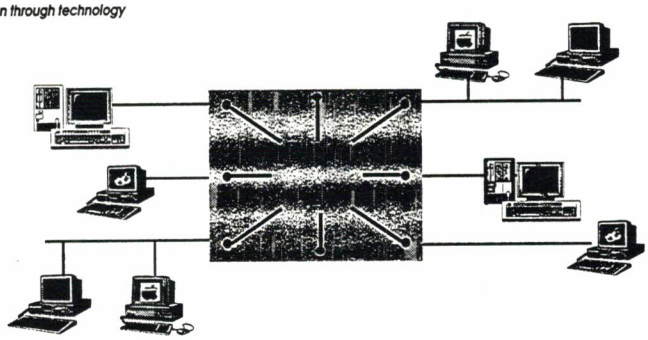
Addressing

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_010A

IBM Networking

Connecting people to information through technology

Switching Issues - Network Analysis



How can you put 'sniffer' on dedicated port?

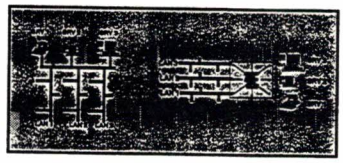
Will your management tools work?

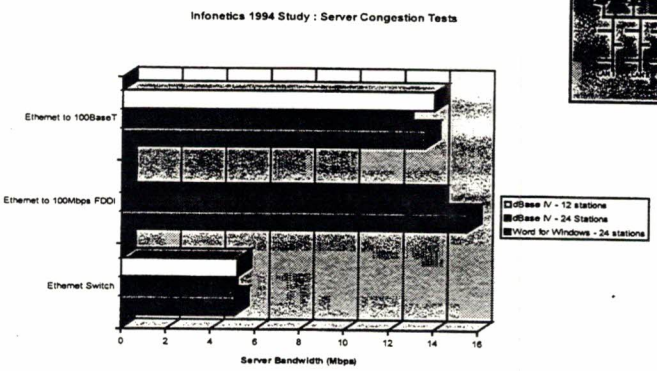
IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 FDX_010F

486 Servers Bandwidth Limitation of 20Mbps

IBM Networking

Connecting people to information through technology



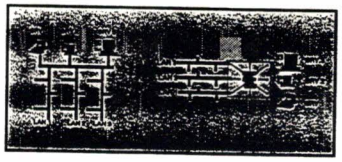


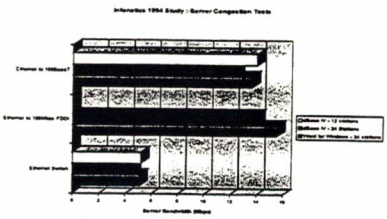
© Copyright IBM Corporation 1993, 1994, 1995

486 Servers Bandwidth Limitation of 20Mbps

IBM Networking

Connecting people to information through technology





Word for Windows - heavy server I/O
dBase - lots of data across the network

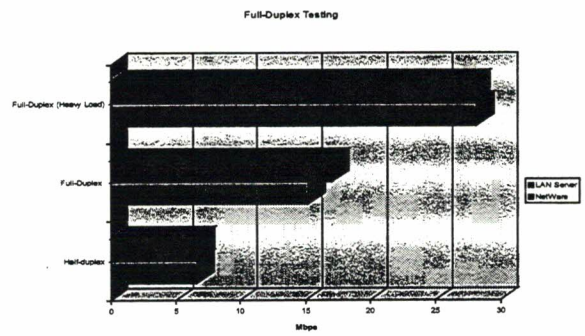
Factors to consider
- **Disk access times**
- **Application design**
- **Bus architecture**
- **Path between systems**

© Copyright IBM Corporation 1993, 1994, 1995

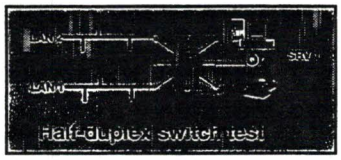
EtherStreamer Full-Duplex Switching Results

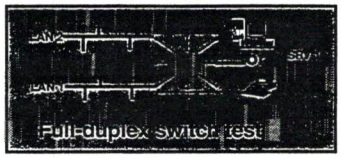
IBM Networking

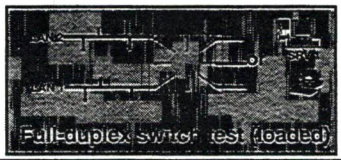
Connecting people to information through technology



Tolly Group - 1994





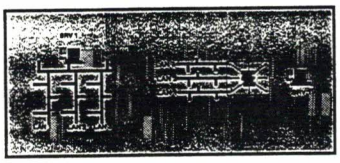


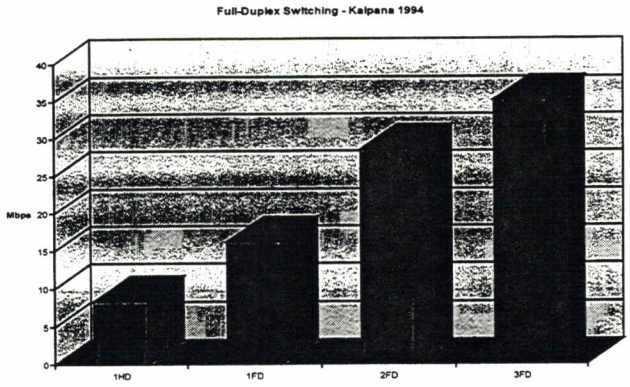
© Copyright IBM Corporation 1993, 1994, 1995

Server Adapter Half/Full Duplex Advantage

IBM Networking

Connecting people to information through technology





Kalpana Testing - 1994
(Multipe FDX requires load balancing software)

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking — Technology Comparisons

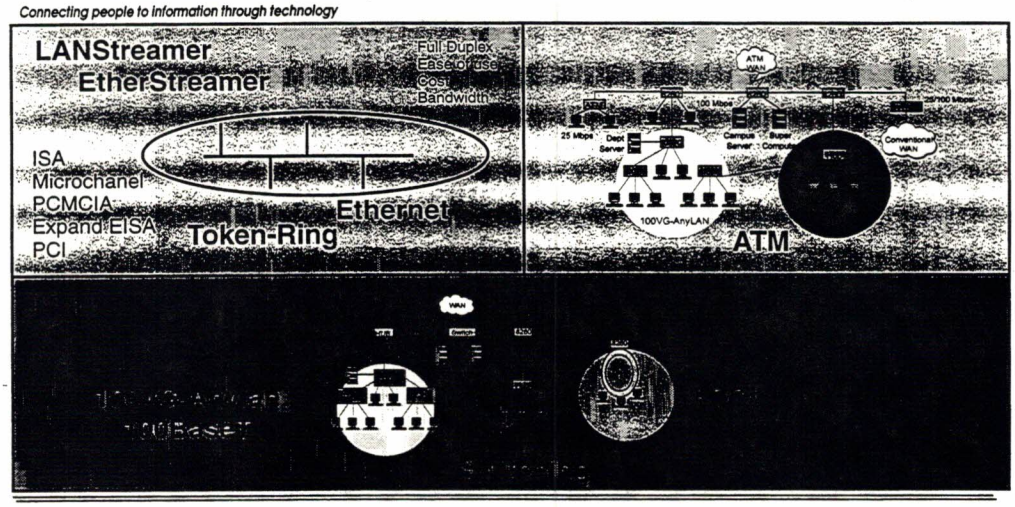
Connecting people to information through technology

Technology Solution	Potential Server Bandwidth	Shared or dedicated	Servers	Equipment costs	Cost to implement	Ready
Switching,	10 Mpbs (ethernet) 16 Mpbs (token-ring)	Dedicated	Low end	Low	Low	Now
Full-duplex switching	20 Mbps (ethernet) 32 Mbps (token-ring)	Dedicated	All	Low	Low	Now
High Speed Lan	100 Mbps	Shared	High end	Moderate	High	Now
ATM	155 Mbps	Dedicated	High end	High	High	Limited

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking — IBM LAN Technology Support

Connecting people to information through technology



© Copyright IBM Corporation 1993, 1994, 1995

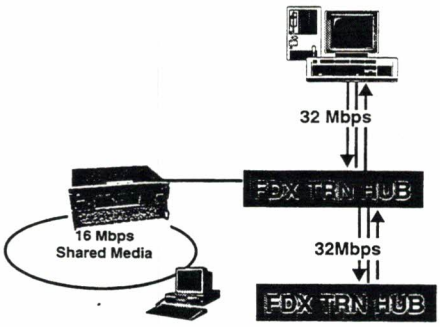
When to Implement Switched LANs

IBM Networking

Connecting people to information through technology

Server Centralization
- Problems
  - backbone congestion
  - increased delays
  - bridge/router congestion
  - decreased response time
- Solutions
  - Upgrade backbone from 4Mbps -16Mbps
  - Use FDX TRN at backbone
  - For long term migrade to ATM backbone

High Bandwidth Local Applications
- Problems
  - segment congestion
  - access delays
  - decreased response time
- Solutions
  - Upgrade from 4Mbps-16Mbps
  - Upgrade to FDX TRN for device
  - Migrate to high speed lan technology
  - Migrate to ATM



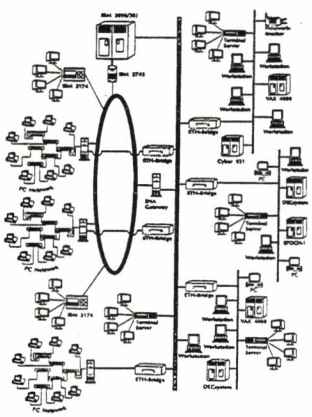
© Copyright IBM Corporation 1993, 1994

Switched LAN Actions Today

IBM Networking

Connecting people to information through technology

What you can do today to provide future growth into multimedia



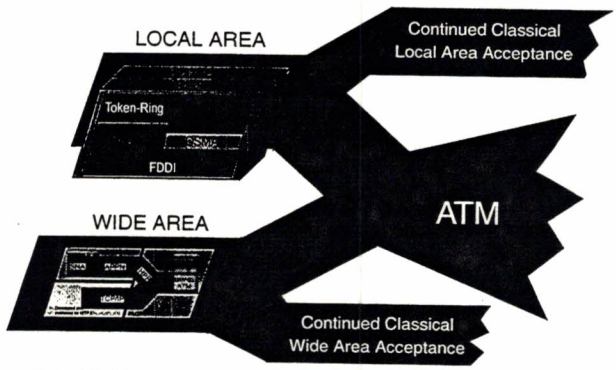
Action	Product
Add high bandwidth adapter to server	Auto LANStreamer is FDX enabled
Migrate to collapsed backbone configuration	Future standalone FDX-TRN Switch/Hub
Cabling upgrades	Use Fiber on backbones UTP Type 5 to WS
Migrate to collapsed backbone configuration	IBM 8250 Existing bridges/routers as interim products

© Copyright IBM Corporation 1994

IBM Networking

Connecting people to information through technology

LAN Technologies in Perspective



Full-Duplex Switched LANs

.... For some a solution ... For others a path !!

© Copyright IBM Corporation 1994

MGT 0702
Setting Up an Internal Support Organization

David Moskowitz
Productivity Solutions
Norristown, Penn.

Setting up and internal support organization for OS/2 users

David Moskowitz
Productivity Solutions

Agenda

- Support: OS/2 versus...
- Help users work
- Train users to solve their own problems
- Anticipate problems
- OS/2 for competitive advantage

- How to remain valuable and invisible
  - Avoid the perception of problems

Perceptions

- If "it" needs constant support...
  - Management will get rid of "it"
  - Won't need people to support "it"
  - "It" is too difficult
  - There might be something wrong with "me"
- If the users hate "it" they aren't going to feel to good about "you" either.

Types of support

- Application suites -- combination of business application, software, and hardware
  - Support people trained for this limited set
  - Advantages -- most successful for narrow area (email, access database) etc. & not tied to specific support personnel
  - Disadvantages -- organization doesn't learn and stays at novice stage; no one expected to do better; the organization can't support it.
  - The real experts (if any) are hidden.

Copyright (c) 1995 David Moskowitz. All rights reserved.

Types of support

- Expert system
  - Filter layers to get to the experts
  - Leveraged into expert people (expensive)
    - Difficult to build additional experts
  - Experts may not find that their time is entire devoted to support
  - Experts can go crazy
  - Psychological dependence on the expert

Types of support

- Value added support group
  - Support group acts as facilitators instead of reacting
  - Combine training and support function in one group
  - OS/2 is new -- need to develop both emotional support and knowledge support.
  - Call back follow up to help reduce need for a support call next time.
  - Integrate training and support, service, hotline, internal consulting, etc.
  - Talk to new OS/2 users and find out what they think they will need.

Support can't be isolated

- Failure to plan is planning to fail
- Involve the support group in the early planning stage
- Support people need people skills and technical skills

- Problems are people problems, not technology problems. People either created the problem or the need for a solution.

Support differences

- DOS or Windows
  - problems usually linear
  - Frustration
    - Configuration, installation
  - Connectivity not a strong point nor is Windows CSA ready
  - GUI
  - OS/2
  - Problems can be non-linear
    - Timing, sharing, intermittent, transient
  - Once installed, minimal problems
  - Good client
  - OUI

Copyright (c) 1995 David Moskowitz. All rights reserved.

Support changes

- Windows question
  - Can I...
- OS/2 question
  - - How can I...

The single word makes a big difference in the way users approach the systems. It also changes the expectations of services that can be delivered by the support organization.

"I used to be able to do that in..."

New Expectations

- You cannot drop new "stuff" on users and expect miracles
- 5 Levels of learning
  - Novice, Expert, Wizard, Warrior, Dungeon Master
- 3 or 4 levels of users
  - Beginner, Experienced, Power user or programmer

Develop users

- User can solve their own problems if you work with them.
- Anticipate problems
  - Keep history of problems
    - Correlate with department, people, management, tools, hardware, systems, etc.
    - Publish the top types of requests that account for 80 percent of calls.
    - Work with development department to modify course content.

Keys to success!

- Encourage users to ask questions
- Make sure they understand that if "it" breaks "I'm" not the cause.

Copyright (c) 1995 David Moskowitz. All rights reserved.

Anticipate problems

- One of the critical issues is people skills.
  - Match user learning mode with support type
    - Show me...
    - Tell me...
    - Let me...
    - Find similarities or differences
  - Listen to determine the need, the thneed and...

OS/2 for Competitive Advantage

- What features can you exploit?
  - Multitasking, stability, Bonus Pak applications, Internet access, etc.
- What benefits can you anticipate?
  - Increased productivity, fewer complaints (after learning curve), creative problem solving, etc.
- How do you measure the return?
  - Waiting time
  - What constitutes, "waiting time?"
  - How much time can you recapture?

Survey example

- Survey waiting time...
  - Asked people how much time they thought they spend waiting.
    - How much time do you think you spend waiting for your computer, as a percentage?
  - Measured waiting time
  - Surprised by the results
  - Repeated this 34 times since original study.

Real life example

- Developers influenced company.
- 4 management levels between CEO & janitor
- Everyone has email
- Everyone shares productivity figures
- Typical curve following training... 90 day productivity higher than original but not as high as 10 days
- Developer Productivity was higher 90 days out than it was a 10 – something was different
- To spread wealth...

Copyright (c) 1995 David Moskowitz. All rights reserved.

Company planned rollout

- Small groups got OS/2 after some hands on work. Initial training by developers.
- Challenge to attendees -- "Treat this information as if you have to teach it to someone else, tomorrow."
  - They did!
- Setup internal mentoring and support groups
- Develop a learning organization

Invisible value

- Object: reduce problems so that they are manageable by the organization not a threat to mission or purpose.
- OS/2 can provide value as long as there is a plan for success.
- Set reasonable expectations.
- Make training and development part of the rollout effort
- Involve people in the process from the beginning

Questions...

- The only dumb question is the one that isn't asked.
- If you don't understand something it is my fault. If you don't ask the question the responsibility shifts to you.
- Want different results, change the questions you ask about the process.

Thank you!

For more information:

David Moskowitz
Productivity Solutions, Inc.
164 Avondale Road
Norristown, PA 19403-2952
Voice: +1-610-631-5685 Fax: +1-610-631-0414
Internet: dmoskowitz@cis.compuserve.com
Internet: davidm2@ibm.net
CompuServe: 76701,100

Copyright (c) 1995 David Moskowitz. All rights reserved.

MGT 0705
15 Steps to Downsizing to OS/2

Robert Chapman
Chapman & Associates
Aliso Viejo, Calif.

INTRODUCTION

Downsizing corporate applications is an exciting topic. With the introduction of OS/2 five years ago it became possible to move some mainframe applications down to the Personal Computer. With the release of OS/2 2.0 in the Spring of 1992 the capacity of the PC finally reached the critical value necessary for large scale computing applications. OS/2 can execute enough processes simultaneously, support enough memory, and access enough disk space, to finally earn the PC the right to run corporate America's small business applications. Increases in hardware capacity have kept up with OS/2. There are phenomenally quick microprocessors on the market today. The quoted disk capacities and data transfer rates are staggering. The number of PC's which can communicate with a PC Server machine are huge.

However, everything is not peaches and cream. These are still the early days of downsizing and an appropriate attitude to take is "Downsizer Beware". It is still possible to create a "small" application which can drive OS/2 to its knees. The intent of this article is to provide you, the average user and developer, with the basic understanding you need to succeed in your downsizing efforts.

MAINFRAME HARDWARE VERSUS PC HARDWARE

Most of the problems with downsizing are related to hardware capacity. Quite often the processing load of the application exceeds the amount of projected hardware requirements. This is caused by people looking at a machines most popular ratings, the microprocessor MIPS (Millions of Instructions Per Second) and the maximum disk transfer rate, as the sole representatives of power. These numbers are misleading and represent a fundamental problem with today's perception of PC performance.

Mainframes, in this article I am using the IBM ES9000 series to represent mainframes, have boring measurement numbers. Mainframe processors look slow compared to the new Intel Pentium processors. For example, the base CPU of an IBM ES9000 might be rated at 13 to 50 MIPS, while a Pentium is rated at from 60 to 100 MIPS. The difference in price is incredible. The mainframe might cost you $3 million while the Pentium will run $6000. Let us take a look at the difference between an IBM mainframe running 50 MIPS and a Intel Pentium running at 60+ MIPS. This will identify the most significant reasons why the 50 MIPS mainframe can run thousands of those "small" applications while the 60+ MIPS Pentium might choke on one.

CHANNELS VERSUS BUS STRUCTURE

Mainframes have Channels which are direct paths to main memory. Each Channel can transfer 10 megabytes (MB) per second between the I/O devices attached to it and the main memory. There can be up to 256 Channels on a mainframe. Aggregate I/O throughput of the mainframe can reach 2560 MB per second.

PC's have a BUS, which is a shared data path , along which data travels between devices (microprocessor to I/O devices, I/O devices to memory, I/O device to I/O device). PC's can have one ISA, EISA, or MCA bus. Many new PC's also have one Local Bus available. The ISA bus can transfer up to 8 MB per second. The standard MCA bus can transfer 20MB per second, but can be pushed, under special circumstances to 160 MB per second (There are currently no devices capable of reaching this rate. Today there are only a few adapter cards which can transfer data at 40 MB per second). The standard EISA bus can transfer 32 MB per second, but theoretically it can be pushed higher (under the same special circumstances as the MCA bus). The Local Bus can transfer up to 132 MB per second. Aggregate throughput of the PC bus runs from 8 MB per second up to a theoretical maximum that exceeds 300 MB per second. Today's PC's rarely reach 8 MB per second.

CHANNELS VERSUS ADAPTER CARDS

Each mainframe Channel can connect to 16 Units and each Unit can connect to 16 Devices. Devices are Disk

drives, printers, readers, telecommunications devices. There can be up to 256 Channels and over 65000 Devices physically connected to a single mainframe. Channels can communicate independently of the CPU and can control multiple Units simultaneously. Channels require communication with the CPU only to start an I/O sequence, and to manage the I/O interrupts. Channels are basically special purpose computers which execute asynchronously with the CPU.

PC's can support a limited number of adapter cards, the maximum number depends on the available slots on the mother board. A typical maximum is 7 adapters. Different types of adapter cards are available. The early disk controller adapter cards could connect to 4 disk drives, two of them floppy disks, two of them hard disks. SCSI adapter cards, the most flexible cards available, can support up to 7 devices. Some asynchronous serial adapters can support over 16 serial devices simultaneously. Using these numbers as a guide, the maximum number of devices that can be attached to a PC can range from 28 to several hundred. In real life though the numbers are much smaller. A typical high end PC is connected to less than 10 devices, and usually cannot service all of those in a timely fashion.

Disk Speeds and Capacities

Mainframe disk drives are generally rated at around 18 milliseconds average access time. A typical mainframe disk drive has multiple head assemblies (up to 6). Each head assembly has two actuators (basically an independent read/write head) which can be accessed (addressed and read/write data) separately from the others. Each head assembly can transfer data at rates exceeding 4 MB per second. Several head assemblies accessed simultaneously can easily fill the 10 MB per second capacity of a single Channel. Mainframe disk drives broke the 1 gigabyte (GB) barrier over 15 years ago. They broke the 100 GB barrier this year (1993). A typical mainframe disk drive today contains 30 GB of data.

PC disk drives are generally rated at around 12 milliseconds for the average access time. A PC disk drive has one head assembly. PC disk drives almost never fill the bus, but can reach bus capacity for short bursts of time. PC disks are available in capacities ranging from 100 MB to 5 GB.

One mainframe disk drive can provide more disk space and transfer data faster than most of today's PC's.

Processor Speeds

Mainframe processors are typically rated from 25 to 50 MIPS. Mainframes come with from one to six processors in a single complex. Mainframe processors typically perform from 8 to 16 MFLOPS (Million FLoating point Operations Per Second), but can come with up to 16 Array Processors executing over 100 MFLOPS each. PC processors range from 10 MIPS all the way up to 60+ MIPS. PC processors typically perform between 5 and 10 MFLOPS. PC's can support several Array Processors at one time with similar MFLOPS ratings as those attached to mainframes.

MIPS is a very misleading indicator of CPU performance. Mainframe MIPS are usually the equivalent of three or more Intel PC MIPS. There is a subtle but important reason for this. Mainframes have instruction sets that were architected to address known computing requirements. Microprocessor instruction sets were originally based on what could be easily etched in silicon and evolved from there. (One of the reasons RISC processors are so successful is that they are architected for performance, they did not evolve into their current form.) Mainframes typically have no wasteful instruction requirements.

Summary

Simply reviewing the base performance and capacity numbers between mainframes and PC's give you some idea of the difference in processing power. A mainframe can support hundreds of times the number of peripheral devices, usually disk drives, and can transfer hundreds of times more data per second between these devices than a PC can. The mainframe instruction set is designed for performance, and the CPU does not have to be con-

cerned with the details of the I/O in progress. Additionally, up to six CPU's can be tightly coupled (for symmetrical multiprocessing or SMP) in a single computer system. When you add it all up you see that a mainframe computer has from 1,000 to 10,000 times the processing capacity of today's PC's.

OPERATING SYSTEMS - MVS VERSUS OS/2

Multiple Virtual Storage (MVS) is IBM's flagship mainframe operating system. Operating system facilities play a major part in system performance. Some operating systems are designed to optimize interactive terminal performance. Some operating systems are designed to optimize batch job processing. Some operating systems are designed to optimize data transmission processing.

Most operating systems can only do one of the above types of processing well. An analogy is fighter airplanes. A fighter can go fast, or it can go far, or it can carry a lot of armament, but it cannot do all of them at once. The same is generally true of operating systems. If one type of processing is to be optimized the other types must be sub-optimized. Alternately, all of the processing types can be sub-optimized so that the operating system can maximize the throughput of the computer system as a whole. In the case of MVS it is much more complicated. MVS attempts to provide "processing type" (on-line, batch, real-time) optimization while still maximizing the overall computer system throughput. OS/2 is designed to provide optimum performance for the foreground application. In OS/2 background applications round-robin through the task scheduler - depending on priority indicators.

In addition to the throughput optimization goals there is the simple fact that the MVS scheduling algorithms have matured over almost twenty years, while OS/2 is less than 6 years old. And, there is more to an operating system than the scheduling facilities.

Communicating Dependencies Between Jobs

MVS has a mature set of batch job scheduling systems available, thus allowing many batch processes to run entirely without intervention. OS/2 has several products that provide scheduling functions, but there is a large difference in magnitude between them and those available for MVS. One example is the magnitude of jobs being managed. It is not unusual for a mainframe to process several hundred batch jobs (jobs consist of one or more programs) each night of the week, and even more on heavy processing nights. The dependencies between each of the jobs can be overwhelming to the novice mainframe programmer.

JOBLIB and STEPLIB versus LIBPATH and PATH

The use of DLL's within OS/2 is a fundamental advance over DOS. But the current OS/2 implementation of DLL's is a major impediment to progress. Executable code can be shared between processes through the use of the DLL facility, but the separation of the DLL search path from the program search path, while providing a small improvement in overall performance, increases the complexity of the OS/2 execution environment. MVS provides access to programs and DLL type capability through the use of the STEPLIB and JOBLIB commands. These provide the same function as the PATH command in OS/2. Unfortunately, the OS/2 LIBPATH command is not like the PATH command. The PATH command can be re-executed for each individual process, but the LIBPATH command is static and is associated with every process. This makes DLL maintenance difficult in a multi-application environment.

System Catalog

The MVS system catalog facility allows applications to be separated from the location and device type of the files which they are processing. This means that single and multiple volume datasets are processed by the application programs with no difference in program logic. Because of this facility, MVS can easily manage hundreds of thousands of datasets..

OS/2 has no built-in system catalog facility, and because of this multi-volume datasets are not supported. It is also harder to manage datasets, because each reference requires the specification of the drive letter and directory path name. This means that in practice every application must know exactly where the required datasets reside before they can be processed. It also means that there is no easy facility for tracking the location of all of the existing datasets. (The OS/2 productivity applet "Seek and Scan" is probably the most frequently used application in OS/2.)

Under MVS it is harder to allocate datasets, because the general space requirements need to be known in advance. Under OS/2 it is harder to utilize datasets because the exact location of each dataset must be known.

Development and Testing Tools

Mainframe development and testing tools are mostly designed to operate in a non-interactive environment. This makes them clumsy and unyielding to use. OS/2 development tools are designed to work in an interactive environment and provide the developer with a large degree of feedback. Without a doubt OS/2 provides a more responsive development environment for the individual. But, the design of the LIBPATH command and the lack of a system catalog facility substantially hinder development efforts. Neither environment exceeds the other in terms of development capacity.

Summary

MVS provides a much more complete execution and development environment for the corporate application. OS/2 provides a more interactive, user friendly, execution environment for the production application. The question of which is more cost effective, MVS and mainframes or OS/2 and PC's, has to be evaluated on a case by case basis.

STEPS TO A SUCCESSFUL DOWNSIZE

Now that the basic attributes of the mainframe (MVS) and PC (OS/2) environments have been covered I have a list of 15 points to follow when downsizing an application. Most of these are pure common sense, but remember the definition of a consultant - "someone you pay good money to say what you already know". Keep these points in mind when downsizing.

The first 10 points pertain to every downsizing effort. The last 5 pertain to downsized applications which are integrated with host applications. By the way, failure to adhere to the first three points listed below probably accounts for 99% of all downsizing problems.

1. First size the hardware requirements

Start the project realizing that the puny 30 MIPS mainframe processor has from 1000 to 10,000 times the processing power of your most powerful PC. Ignore the published performance specifications of the PC platforms and build a benchmark model of your application. Compare the results you get from the different PC platforms under consideration and choose the PC platform with the most "headroom" you can afford. Remember that even if you don't need it right now, the next release of your application will start eating into your excess capacity.

I know one company that had an application requiring a complete database refresh every 30 days, unfortunately their database design caused the database to take over 30 days to load. A quick benchmark would have identified this problem very early in the project.

2. USE YOUR EXISTING ANALYSTS AND PROGRAMMERS

No one needs much PC expertise on a downsizing project. Keep in mind that PC experts which have not come from the mainframe environment have little appreciation for the constraints of a corporate business system. Existing mainframe MIS staff can easily learn how to program and design for the PC environment.

3. USE THE SAME LANGUAGES THAT YOU USE ON THE MAINFRAME

The maintenance costs for any downsized system are going to exceed the development cost by orders of magnitude as the years go by. All of the major languages are available (and stable) on OS/2. If your programmers use COBOL, then your downsized application should be developed in COBOL. The same rule applies to FORTRAN and C. It is not unusual for a company to change languages for their PC development and then find that they cannot keep the maintenance positions staffed.

4. USE MAINFRAME ARCHITECTURES WHEREVER POSSIBLE

For example, now that IBM has released full function copies of CICS and DB2 for OS/2, your applications can maintain the same structure, and utilize existing support libraries from the mainframe. Many other companies have also migrated their products to OS/2. Additional user friendly features can be added to those downsized applications as the project team gets familiar with building GUI applications. Be smart and start with what you know.

5. USE YOUR EXISTING DEVELOPMENT METHODOLOGIES

Many people say that development is different under OS/2 than on the mainframe and that the strict rules and procedures of the MIS environment do more to hinder the downsizing effort then help. My experience is totally the opposite. Development methodologies and procedures allow the development team to stay on course throughout the life of project. Throw these out at your own risk.

6. DO NOT REINVENT EXISTING TECHNOLOGIES.

Buy what you can, build only what you cannot buy. There are three reasons for adopting this philosophy. First, the maintenance costs of anything you build are going to substantially exceed your development cost. Second, mature products almost always have more and better features than you can build in the short amount of time available to you. Third, the purchase price, even when extended over multiple systems is usually lower than your own internal development costs. However, always keep point number (1) above in mind when reviewing commercial software.

7. GET A REMOTE CONTROL PACKAGE FOR OS/2

Most downsized applications will be executing on a LAN, and most will not be local. Realize from the beginning that you will be downloading via communications links both programs and data files. Make sure that your design takes into account the remote maintenance requirements of the application.

8. GET A SOURCE MANAGEMENT PACKAGE FOR OS/2

Without a source management package your OS/2 development effort will become a mass of anarchy. This will lead to increased costs and decreased reliability. Any source manager is better than none. You cannot run a project without a source manager, regardless of how intelligent or well meaning your team members are.

9. Get an automated testing tool

You can not trust your programmers to adequately test their programs in a systems environment. Get the tool before you start coding and have your development staff use it for everything - including their own unit tests.

10. Use PC development tools ONLY if they will not cause maintenance problems

Many development tools lock the developer into a proprietary environment. Using PC development tools which only give you a short term gain is a fools game. Remember that any system you downsize will be around for years and will be undergoing constant refurbishment. Always think about the maintenance cost of the decisions you make.

IMPORTANT POINTS FOR HOST CONNECTED APPLICATIONS

1. Remember the communications link

Any OS/2 application which is connected to a mainframe is immediately presented with a bottleneck device - the communications line. Make sure that your application can be adequately served by the communications line.

2. Move as little data as possible between the mainframe and OS/2

Data moving between the mainframe and OS/2 travels up to 1,000 times slower than within the mainframe or the PC alone. Communications links are much less stable than the other technologies in a downsized system. Whenever possible your application should be designed to minimize the amount of data moving between the mainframe and OS/2, and the scheduling dependencies between the mainframe and OS/2.

3. Identify, understand, and minimize any mainframe to OS/2 dependencies

There are different scales of time on mainframes and PC's. It is not unusual to move thousands of megabytes of data per hour on a mainframe, but this is very unusual on a PC. So, when a mainframe application has a one hour window in which a process must complete, keep in mind that an OS/2 process might not be able to finish within that window. This time scale difference can really complicate the design and execution of a downsized application.

One application I saw needed to move 750MB between the OS/2 application and the mainframe application within a 3 hour window. It was just barely possible, but only with thoughtful design work. A quick implementation, without consideration of the effect of the volumes of data involved, or the synchronizing requirements of the mainframe and OS/2 applications, would have resulted in disaster.

4. Always use transparent mainframe to PC link technology

This is a variation of point (6) above. If a product is on the market which allows you to move data from OS/2 to the mainframe without your applications being aware of the source or target of the transfer, use it. There are several products on the market today which provide this. Even if those products have faults, they are better than anything the average development team can produce in the time allotted to them.

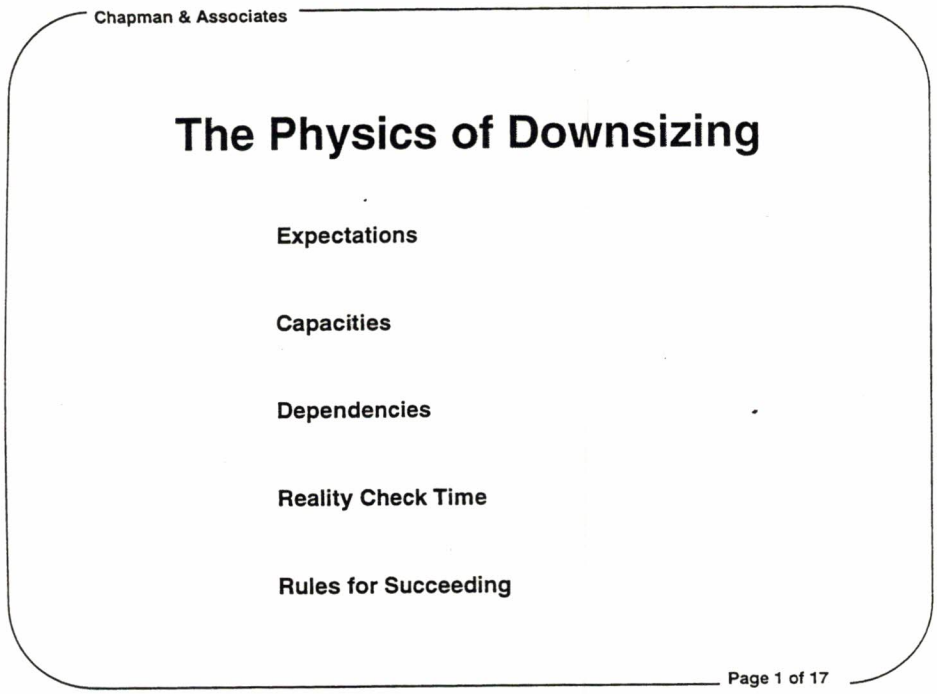
5. Don't shy away from home grown APPC links to your mainframe

But do not build them if you can buy them. Using IBM's Systems Applications Architecture (SAA) Common Programming Interface - Communications (CPI-C) technology any programmer can create an APPC link from

OS/2 to the mainframe. Before you take this step make sure that the communications link is adequate for the job you are proposing.

The 15 points identified above are only common sense, and most people already know them. When the downsizing development effort starts it is sometimes hard to keep everything in perspective. If you do, you will have no trouble downsizing any application which can fit on your OS/2 system.

One last comment, as cute as that PC on your desk is - it is not the solution to all of your mainframe problems



Chapman & Associates

Expectations

Low Cost

Powerful User Interface

Quick Response Time

Easy Implementation

Fun

Page 2 of 17

Chapman & Associates

Capacities

CPU and I/O Power

Disk Speed and Capacity

Operating System Features

Page 3 of 17

Chapman & Associates

CPU and I/O Power

	IBM ES9000	Intel Pentium
MIPS per Processor	30	60-100
Number of Processors	6	1-4
Maximum Memory	8 GB	4GB
Paths to Memory	6	1
Channel Capacity	10 MBS	10 MBS
Number of Channels	256	10
Paths to Channels	256	4
Devices Per Channel	256	56

2 x 6 x 25.6 x 64 x 4.5 = 89,878 times the CPU/IO Power

Page 4 of 17

Chapman & Associates

Disk Speed and Capacity

	IBM 3990	SCSI-2
Average Access Time	18 ms	15ms
Maximum Xfer Rate	4.2MBS	5 MBS
Head Assemblies	12	1
Simultaneous Access	Yes	No
Physical I/Os Per Second	50	50
Maximum Capacity	100GB	5GB
Average Capacity	30GB	1GB

(12 x 30) / (1.2 x 1.2) = 250 Speed/Capacity Times More

Page 5 of 17

Operating System Features

	MVS	OS/2	DOS
Scheduling Algorithm	Throughput	Foreground	None
Access Methods	Yes	No	No
System Catalog	Yes	No	No
Scheduling Packages	Yes	Yes	No
Development Tools	Yes	Yes	Yes

Dependencies

Job Dependencies

File Dependencies

Host File Dependencies

Windows of Opportunities

Job Dependencies

PC's Do Not Have Job Concept

Job's Would Be Collections of Programs Held Together By REXX Procedures

As Application Systems Proliferate Interdepencies Will Increase

Increasing Subsystems Increase Job Management

File Dependencies

Online Files

Batch Files

Database Files

Audit Logs

Transaction Files

Host Dependencies

File Availability

Communications Bottleneck

Scheduling Considerations -

- Host Initiates PC Processing
- PC Initiates Host Processing

Windows Dependencies

Processing Windows

1 Hour of Host Does Not Equal 1 Hour of PC

Signaling Processing Status

Chapman & Associates

Reality Check Time

Hosts Are 1000 to 10000 Times More Powerful

Hosts Process Data 10 Times Faster

Hosts Have Experienced Production Staff

PC's Are Neat!

PC's Have GUI Interfaces

PC's Are Controlled Locally

Page 12 of 17

Chapman & Associates

15 Rules For Succeeding

1. Size the Hardware Requirements
2. Use Existing Analysts and Programmers
3. Use Current Languages (COBOL, FORTRAN, etc...)
4. Use Mainframe Architectures
5. Use Existing Development Methodologies

Page 13 of 17

15 Rules (Continued)

6. Do Not Reinvent Existing Technologies
7. Get A Remote Control Package
8. Get A Source Management Package
9. Get An Automated Testing Tool
10. Use PC Development Tools ONLY If They Will Not Cause Maintenance Problems

15 Rules (Continued)

Host Connected Systems

11. Remember The Communications Link
12. Move As Little Data As Possible
13. Identify, Understand, And Minimize Mainframe Dependencies
14. Always Use Transparent Mainframe to PC Links
15. Don't Shy Away From Home Grown APPC Links

Summary

PC's Are Not Always More Cost Effective

PC's Are Not Always Powerful Enough

PC Development Is Not Cheaper Than Mainframe Development

Remember The Maintenance Costs

PC Operating Costs May Not Be Less Than Mainframe Costs, Just Distributed Or Hidden

Summary (Continued)

Make Sure That You Size The Hardware Correctly

Use Your Existing Staff

Use Your Existing Languages

Enjoy Yourself

MM 0708
Multimedia Creation and Management

Peter Yanker
IBM

[RJ9949, Submitted to IEEE Computer - special issue on Content Based Picture Retrieval Systems]

Query by Image and Video Content: The QBIC system

Myron Flickner Harpreet Sawhney Wayne Niblack
Jon Ashley Qian Huang Byron Dom
Monika Gorkani Jim Hafner Denis Lee
Dragutin Petkovic David Steele Peter Yanker

IBM Almaden Research Center
K54/802, 650 Harry Road, San Jose, CA 95120-6099
email contact: flick@almaden.ibm.com
phone (408) 927-1776 fax: (408) 927-4020

keywords: image/picture/video database, content based image/picture/video query/retrieval/annotation/indexing

Abstract

Advances in scanning, networking, compression and video technology, and the proliferation of multimedia computers have led to the generation of large on-line collections of images and videos, creating a need for new methods to locate specific images or video clips. The Query By Image Content (QBIC) project is studying methods to extend and complement text-based retrievals by querying and retrieving images and videos by content. Queries can be done using attributes such as the colors, textures, shapes, and object position. Video specific queries include queries on camera motion parameters, i.e., zoom, pan, and object motion. The work has resulted in a prototype system with two major steps: database population and query. In population, methods are available to identify objects in still images, segment videos into short sequences called shots, and to compute features describing color, texture, shape, position or motion information. In database query, images and shots can be retrieved by example (show me images similar to this image) or by selecting properties from pickers such as a color wheel, a sketched shape, a list of camera motions or a combination of these. Key technical issues in QBIC include a visual query language and a graphical user interface (GUI) which allows a user to form a query by painting, sketching or selecting graphical elements; indexing techniques for the high dimensional features describing image and video content; automatic segmentation techniques for image (to identify interesting objects) and video (to identify shots, interesting moving and static objects); and similarity retrieval to match human perception.

1 Introduction

Picture yourself as a fashion designer needing images of fabrics with a particular mixture of colors; a museum cataloger looking for artifacts of a particular shape and textured pattern; or a movie producer needing a video clip of a red car-like object moving from right to left with the camera zooming. How do you find these images? Even though today's technology enables us to acquire, manipulate, transmit, and store vast on-line image and video collections, the search methodologies used to find pictorial information are still limited. Typically, they depend on file id, keywords or text associated with the images, and although powerful, do not allow queries based directly on the visual properties of the images, are dependent on the particular vocabulary used, and do not provide queries for images "like" a given image. Research on ways to extend and improve query methods for image databases is widespread, and results have been presented in workshops [6], conferences [5, 10], and surveys [4].

We have developed the QBIC (Query By Image Content) system to explore content based retrieval methods. QBIC allows queries on large image and video databases based on example images, user constructed sketches and drawings, selected color and texture patterns, camera and object motion, and other graphical information. Two key properties of QBIC are (i) its use of image and video content – computable properties of color, texture, shape, and motion of images, videos, and their objects – in the queries, and (ii) its graphical query language in which queries are posed by drawing, selecting, and other graphical means. Related systems, such as MIT's Photobook [11] and the Trademark and Art Museum applications from ETL [7], also address these common issues. This article describes the QBIC system and demonstrates its query capabilities.

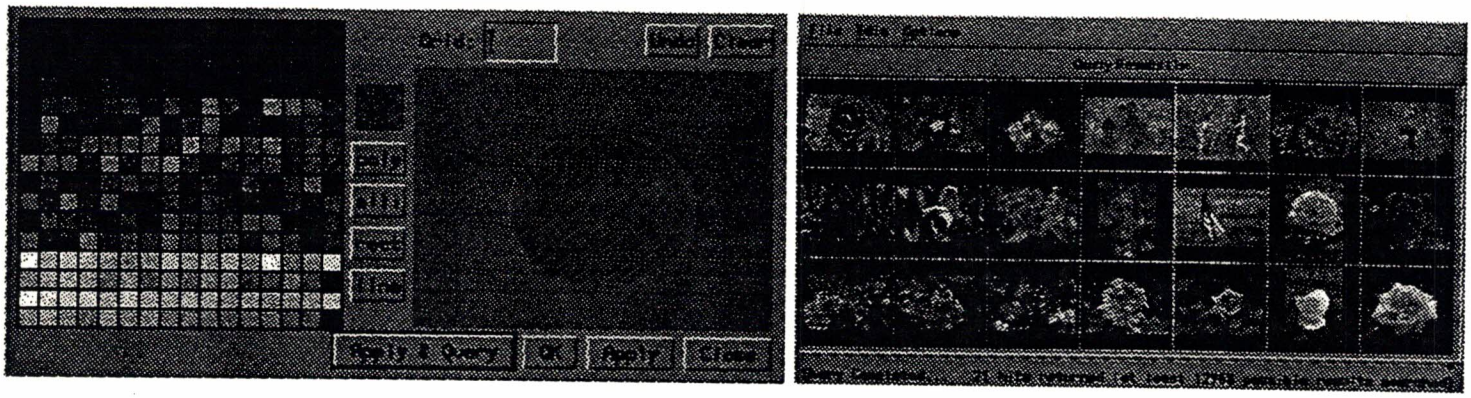


Figure 1: QBIC query by drawn color. Drawn query specification on left. Top 21 results from a 12968 picture database on right.

2 Overview of the QBIC system

A typical query [1] in QBIC is shown in Figure 1. The left side shows the query specification, where the user painted a large red circular area on a green background using standard drawing tools. Query results are shown on the right; an ordered list of "hits" similar to the query. The order of the results is top to bottom then left to right to support horizontal scrolling. In general, all queries follow this model in that the query is specified using graphical means – drawing, selecting from a color wheel, selecting a sample image and so on – and results are displayed as an ordered set of images. To achieve this functionality, QBIC has two main components: database population and database query. During population, images and videos are processed to extract features describing their content – colors, textures, shapes, camera and object motion – and these are stored in a database. During query, the user composes a query graphically. Features are generated from the graphical query, and these are input to a matching engine which finds images or videos

[1] The scene image database used in the figures consists of $\approx$ 7450 images from the Mediasource Series of images and audio from Applied Optical Media Corp, $\approx$ 4100 image from the PhotoDisc sampler CD , $\approx$ 950 image from the Corel Professional Photo CD collection and $\approx$ 450 images from an IBM collection.

from the database with similar features. The system architecture is shown in Figure 2. For both population and query, QBIC has a simple data model: scenes (full images) that contain objects, (subsets of an image), and for video, shots (a set of contiguous frames) and motion objects. For still images, the QBIC data model distinguishes between "scenes" (or images) and "objects". A scene is an image or single representative frame of video. An object is a part of a scene, for example, the fox in Figure 4 or a moving entity in a video. For still image database population, features are extracted from images and objects and stored in a database as shown in the top left part of Figure 2. Videos are broken into clips called shots. Representative frames or r-frames are generated for each extracted shot. R-frames are treated as still images and features are extracted and stored in the database. Further processing of shots generates motion objects, for example a car moving across the screen. Queries are allowed on objects "Find images with a red, round object", scenes "Find images that overall have 30% red and 15% blue", shots "Find all shots panning from left to right", or any combination "Find images that have 30% red and contain a blue textured object".

In QBIC, similarity queries are done against the database of pre-extracted features using distance functions between the features. The distance functions are intended to mimic human perception in order to approximate a perceptual ordering of the database. The collection of all the distance functions is the match engine shown in Figure 2. The match engine interacts with a filtering/indexing module (see sidebar 7) to support fast searching methodologies such as indexing. Users interact with the query interface to generate a query specification, resulting in the features that define the query.

3 Database Population

In still image database population, the images are reduced to a standard sized icon called a "thumbnail" and annotated with any available text information. Object identification is an optional but key part of this step. It allows a user

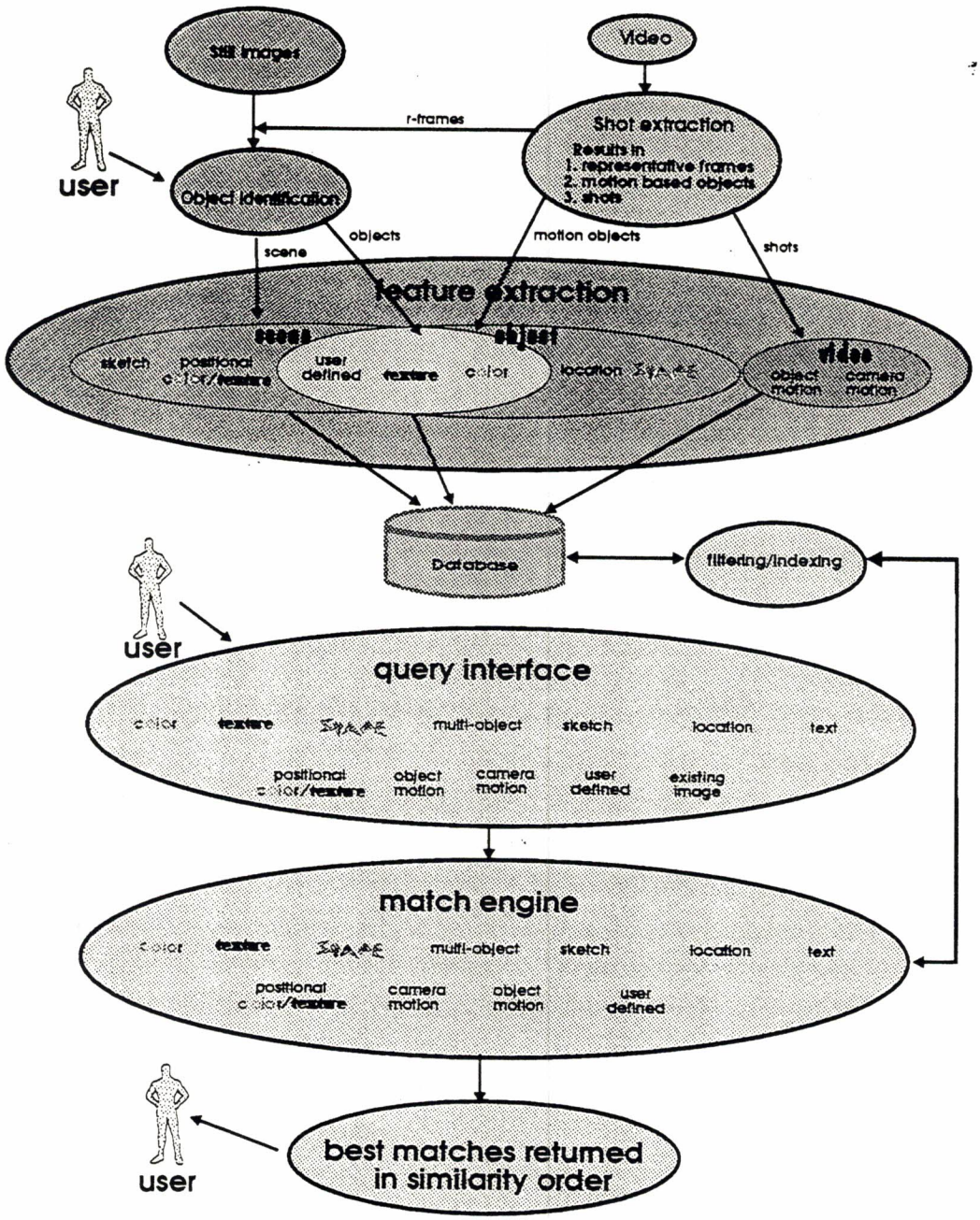


Figure 2: QBIC database population (top) and query (bottom) architecture.

to manually, semi-automatically, or fully automatically identify interesting regions – what we call objects – in the images. Internally, each object is represented as a binary mask. There may be an arbitrary number of objects per image. Objects may overlap and may consist of multiple disconnected components, the set of dots on a polka dot dress, for instance. Text, "baby on beach" for example, can be associated with an outlined object, or to the scene as a whole.

Ideally object identification would be automatic, but this is difficult in general. The alternative, manual identification, is tedious, and can be a inhibitor to query-by-content applications. As a result, we have devoted considerable effort to develop tools to aid in this step. In recent work, we have successfully used fully automatic unsupervised segmentation methods along with a foreground/background model to identify objects in a restricted class of images. The images, typical of museums and retail catalogs, have a small number of foreground objects on a generally separable background. Examples of results are shown in Figure 3. Even in this domain, robust algorithms are required because of the textured and variegated backgrounds.

We also provide semi-automatic tools for identifying objects. One is an enhanced flood fill technique. Flood fill methods, found in most photo editing programs, start from a single object pixel, and repeatedly add adjacent pixels whose values are within some given threshold of the original pixel. Selecting the threshold, which needs to change from image to image and object to object, is tedious. We automatically calculate a dynamic threshold by having the user click on background as well as object points. For reasonably uniform objects that are distinct from the background, this operation allows fast object identification without manually adjusting a threshold. The example in Figure 4 shows an object, a fox, identified using only a few clicks.

Another outlining tool is designed to help a user track object edges. The tool takes a user-drawn curve and automatically aligns it with nearby image edges. Based on "snakes", the tool finds the curve that maximizes the image gradient magnitude along the curve. The spline snake formulation we use

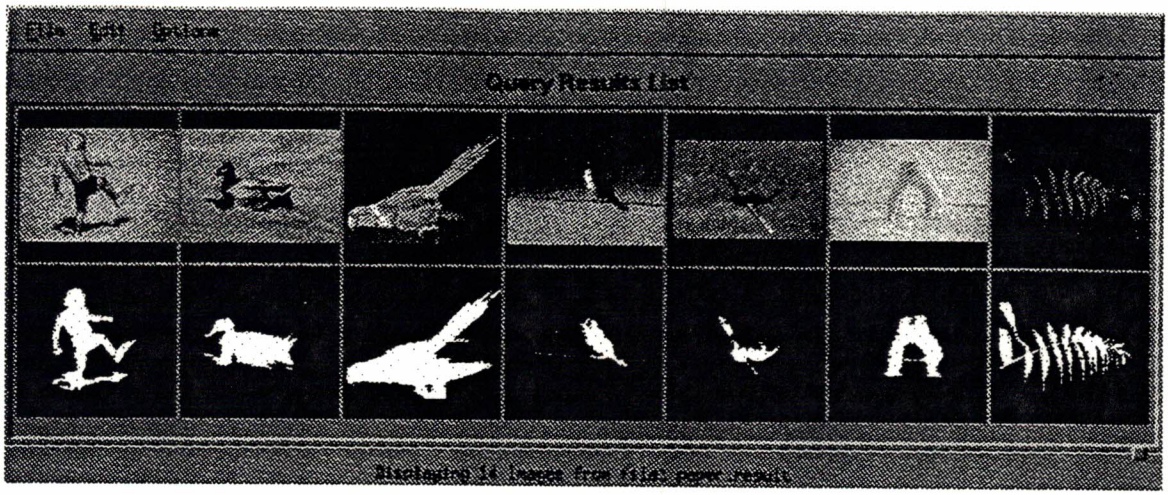


Figure 3: Top row is the original image. Bottom row is the extracted objects using a foreground background model. Heuristics encode the knowledge that object tend to be in the center of the picture.

allows for smooth solutions to the resulting non-linear minimization problem. The computation is done at interactive speeds, so that as the user draws a curve, it is "rubber-banded" to lie along object boundaries.

For video data, database population has three major components: (i) shot detection, (ii) representative frame creation for each shot, and (iii) derivation of a layered representation of coherently moving structures/objects. Shots are small sequences of frames that we use for annotation and querying. For instance, a video clip may consist of a shot smoothly panning over the skyline of San Francisco, switching to a panning shot of the Bay meeting the ocean and then a shot that zooms to the Golden Gate Bridge. In general, a set of contiguous frames may be grouped into a shot because they: (i) depict the same scene, (ii) signify a single camera operation, (iii) contain a distinct event or an action, for instance, significant presence and persistence of an object, or (iv) are chosen as a single indexable entity by the user. Our effort is to detect most of these automatically and provide an easy-to-use interface for the rest.

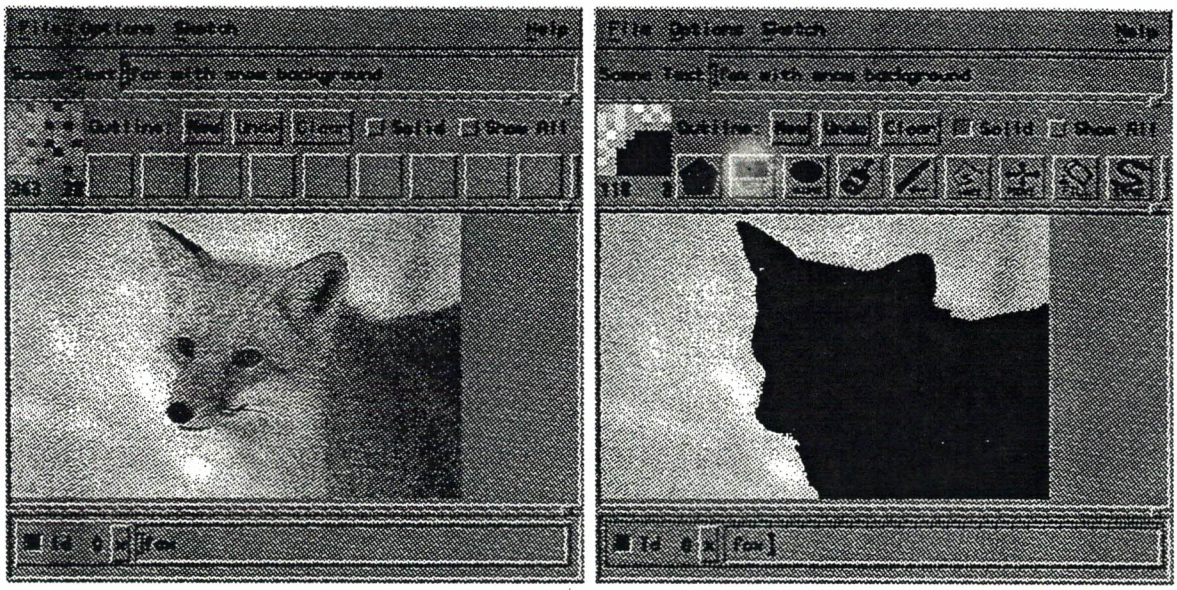


Figure 4: QBIC still image population interface. Entry for scene text at top. Tools in row are polygon outliner, rectangle outliner, ellipse outliner, paintbrush, eraser, line drawing, translate object, flood fill, snake outliner, automatic object finder.

Shot detection: Gross scene changes or scene cuts are the first indicators of shot boundaries. Various methods for detecting scene cuts proposed in the literature essentially fall into two classes: (i) those based on global representations like color/intensity histograms without any spatial information, and (ii) based on measuring differences between spatially registered features, for instance intensity differences. The former are relatively insensitive to motion but can miss cuts where scenes look quite different but have similar distributions. The latter are sensitive to moving objects and camera. We have developed a method that combines the strengths of the two class of methods. Our method uses a robust normalized correlation measure that allows for small motions and combines this with a histogram distance measure [8]. Results on a few videos containing from 2000-5000 frames show no misses and only a few false cuts. Algorithms for signalling edit effects like fades and dissolves are under development. The results of cut detection on a video containing commercial advertisement clips are shown in Figure 6.

Shots may also be detected by finding changes in camera operation. Common camera transformations like zoom, pan and illumination changes can be modeled as unknown affine transformations of the 2D image coordinate system and of the image intensities themselves. We have developed an algorithm [12] for computing the dominant global view transformation while remaining insensitive to non-global changes resulting from independently moving objects and local brightness changes. The affine transformations that are a result of this computation can be used both for camera operation detection, shot boundary detection based on the camera operation and for the creation of a synthetic *r-frame* wherever appropriate.

Shot boundaries can also be defined on the basis of events: appearance/disappearance of an object, distinct change in the motion of an object, or other similar events. For instance, segmenting an object of interest, defined based on its appearance and/or motion, and tracking it throughout its significant presence may be used for defining shots.

Representative frame generation: Once the shot boundaries have been

detected, each shot is represented using an *r-frame*. *R-frames* are used for several purposes. First, during database population, *r-frame*'s are treated as still images in which objects can be identified using methods previously described. Secondly, during query, they are the basic units initially returned in a video query. For example, in a query for shots that are dominantly red, a set of *r-frame*'s will be displayed. To see the actual video shot, the user clicks on the displayed *r-frame* icon.

The choice of an *r-frame* could be as simple as a particular frame in the shot: the first, the last or the middle. However, in situations such as a long panning shot, no single frame may be representative of the entire shot. We use a synthesized *r-frame* [1,13] created by seamlessly mosaicing all the frames in a given shot using the computed motion transformation of the dominant background. This frame is an authentic depiction of all the background that was captured in the whole shot. Any foreground object can be superimposed on the background to create a single, static visual representation of the shot. The *r-frame* mosaicing is done using warping transforms that are the result of automatic dominant motion computation. Given a video sequence with dominant motion and moving object(s), the 2D motion estimation algorithm is applied between consecutive pair of frames. Then, a reference frame is chosen, and all the frames are warped into the coordinate system of the reference frame to create the mosaiced *r-frame*.

Mosaic based *r-frame* creation is illustrated on a video sequence of an IBM commercial. Two frames of this sequence are shown in Figure 5. A single frame dominant-component-only mosaic of the *charlie* sequence is shown in Figure 5. The moving object has been removed from the mosaic using temporal median filtering on the frames in the shot. We are also developing methods to visually represent the object motion in the *r-frame*.

Layered Representation: In order to facilitate automatic segmentation of independently moving objects and significant structures, we take further advantage of the time-varying nature of video data to derive what is called a layered representation [14] of video. The different layers are used to identify

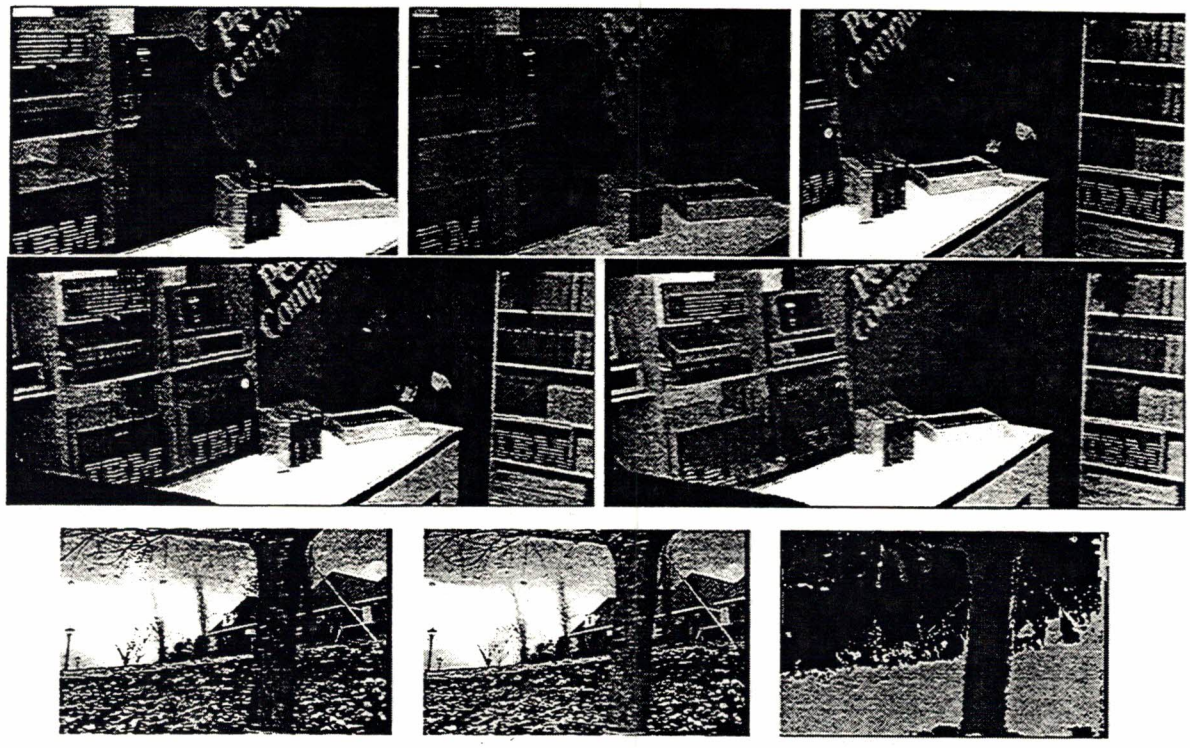


Figure 5: Top: Three frames of the *charlie* sequence and the resulting dynamic mosaics (Middle) of the entire sequence. Bottom: Original images and segmented motion layers for the flower garden sequence in which only the camera is moving. The flower bed, the tree and the background have been separated into three layers shown in different shades of gray.

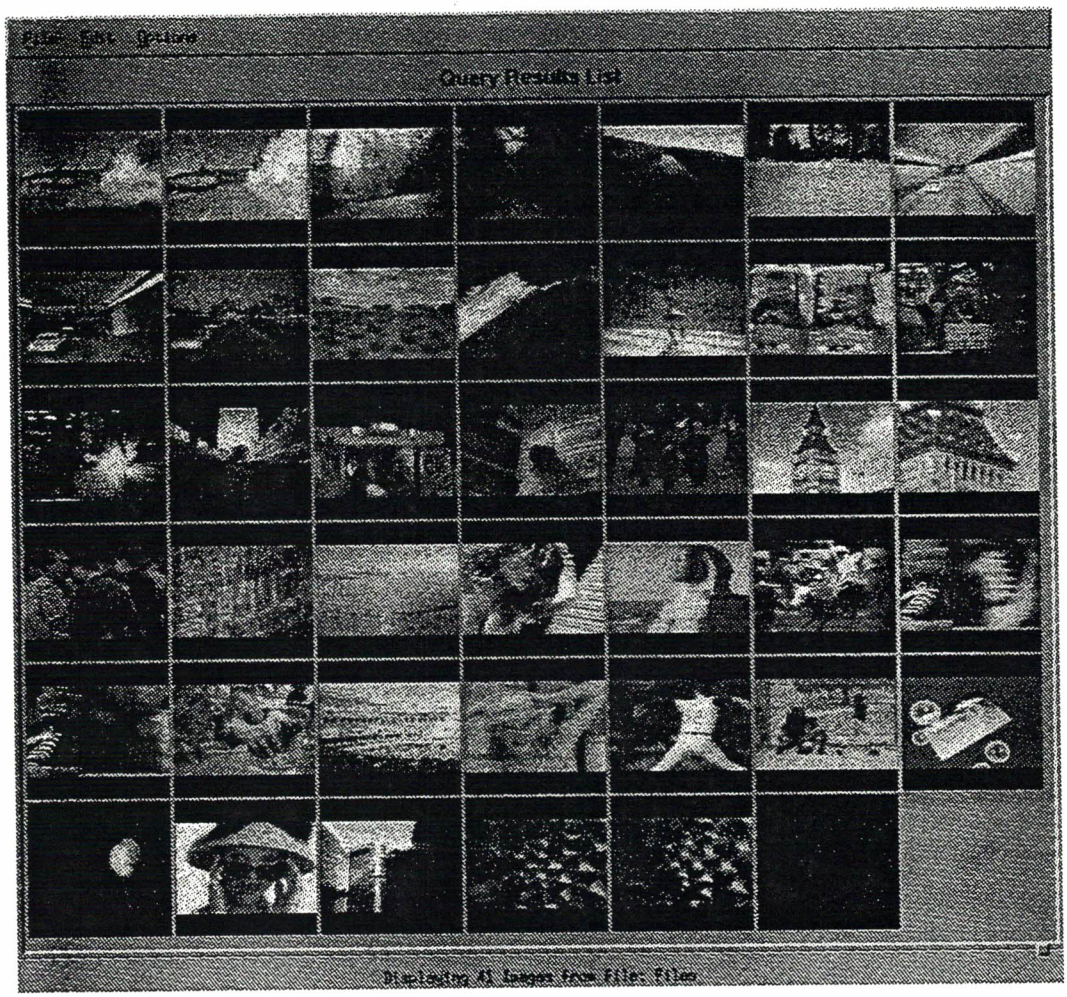


Figure 6: Top: Scene cuts automatically extracted from a 1148 frame sales demo from Energy Productions.

significant objects in the scene for feature computations and querying. Our algorithm divides a shot into a number of layers each with its own 2D affine motion parameters and region of support in each frame [3]. The algorithm is first illustrated on a shot where the scene is static but the camera motion induces parallax motion onto the image plane due to the different depths in the scene. Therefore, surfaces and objects which may correspond to semantically useful entities can be segmented based on the coherence of their motion. Figure 5 shows the results for the layers from the well known flower garden sequence.

4 Sample Queries

For each full scene image, identified image object, *r-frame*, and identified video object resulting from the above processing, a set of features is computed to allow the content-based queries. The features are computed and stored during database population. A brief description of the features together with the associated queries is presented here. Mathematical details on the features and matching methods can be found in [2, 9].

Average color queries allow images or objects to be found that are similar to a selected color, say from a color wheel, or to the color of an object. The feature used in the query is a 3D vector of Munsell color coordinates. Histogram color queries return items with matching color distributions – say a fabric pattern with approximately 40% red and 20% blue. For this case, the underlying feature is a 256 element histogram computed over a quantized version of the color space. Figure 7 shows a histogram query on still images and a color query on video *r-frames*. Notice in the query specification for the histogram query of Figure 7, the user has selected percentages of two colors (blue and white) by adjusting sliders. Using such a query, an advertising agent could, for example, search for a picture of a beach scene, one predominantly blue (for sky and water) and white (for sand), or find images with similar color spreads for a uniform ad campaign. The color

query demonstrates a query against a video shot database where the user is searching for red *r-frames*. Again, the query specification is on the left and the best hits on the right.

An example texture query is shown in Figure 8. In this case the query is specified by selecting from a sampler — a set of pre-stored example images. The underlying texture features are mathematical representations of coarseness, contrast, and directionality features. Coarseness measures the scale of a texture (pebbles vs. boulders), contrast describes its vividness, and directionality describes whether it has a favored direction (like grass) or not (like a smooth object).

An object shape query is shown in Figure 8. In this case the query specification is the drawn shape on the left. Area, circularity, eccentricity, major-axis direction, features derived from the object moments, and a set of tangent angles around the object perimeter are the features used to characterize and match shapes.

Figure 9 illustrates query by sketch. In this case, the query specification is a freehand drawing of the dominant lines and edges in the image. The sketch feature is an automatically extracted reduced-resolution "edge map". Matching is done using a template matching technique.

A multi-object query where we ask for images that contain both a red round object and a green textured object is shown in the bottom of Figure 9. The features are the standard color and texture features. The matching is done by combining the texture and color distances. Combining distances is applied to arbitrary sets of objects and features to implement logical "and" semantics.

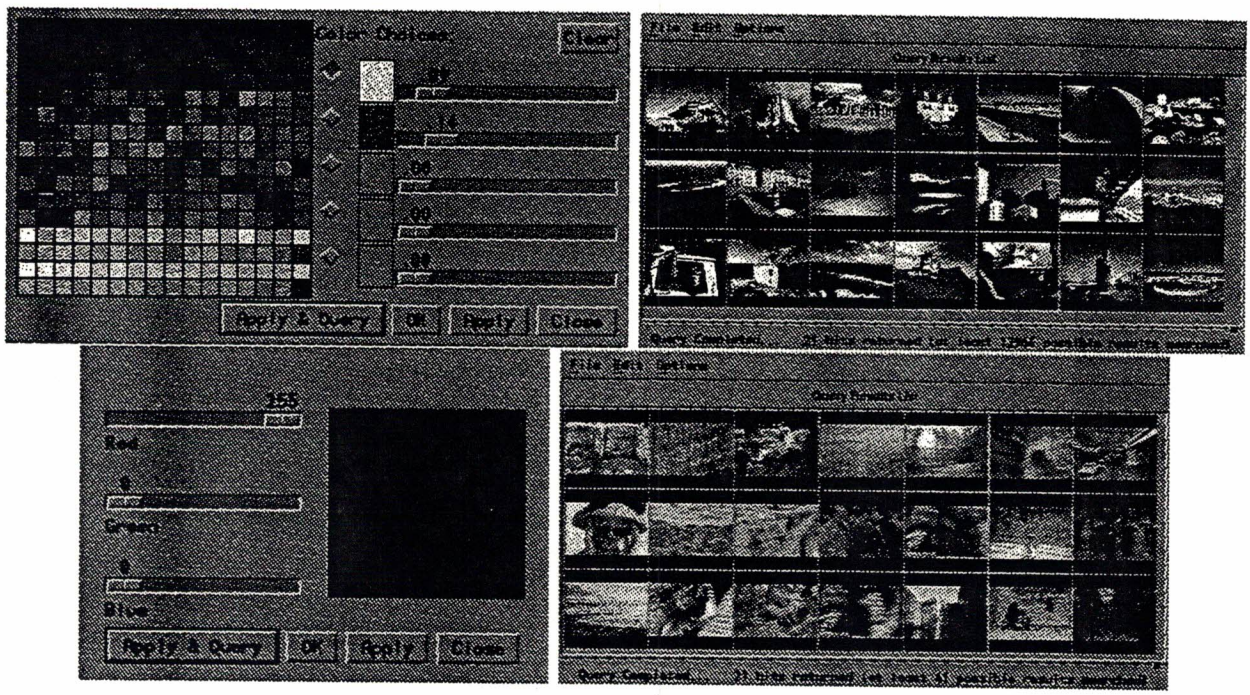


Figure 7: Top : Query by histogram color. Histogram color query specification on left. Best 21 results from a 12966 picture database on right. Bottom: A query for a red video *r-frame*. The color picker is on the left. The resulting *r-frame* thumbnails of the best matches are shown on the right. Each thumbnail is an active button that allow the user to play the shot.

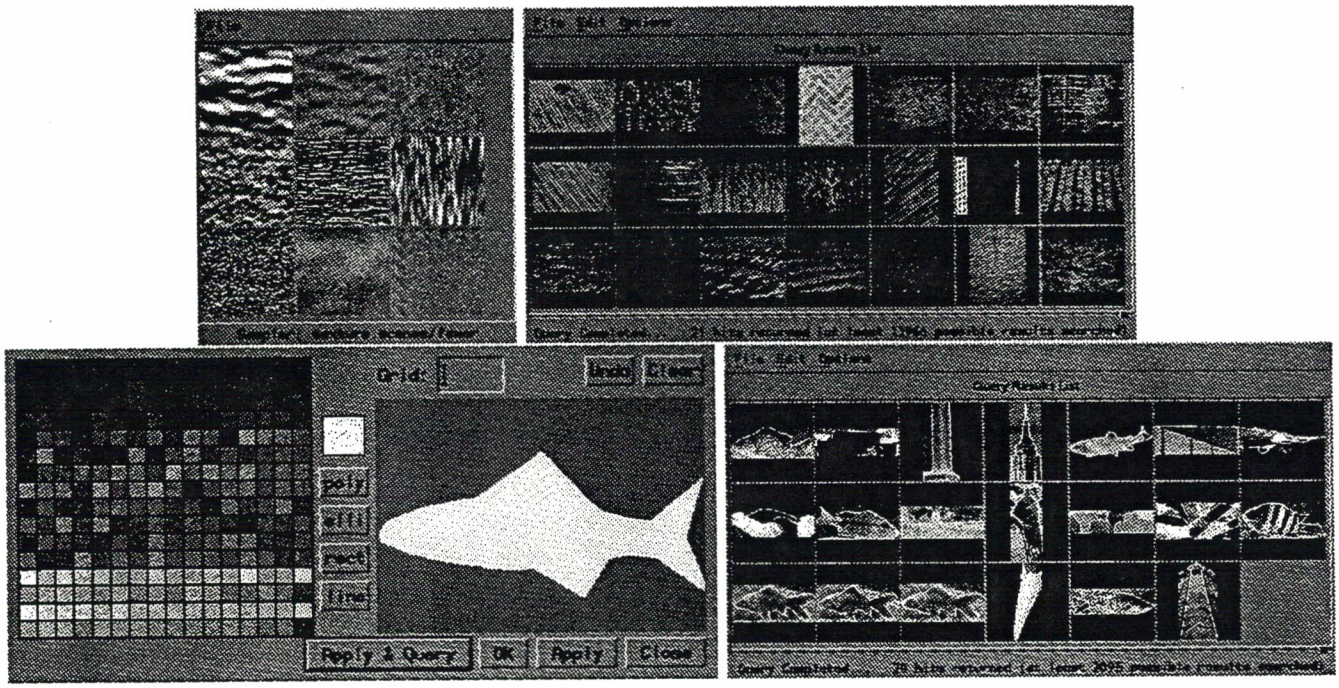


Figure 8: Top: Query by texture. Texture sampler on left. The right middle texture was the query specification. Top 21 results from a 12966 picture database on right. Bottom: Query by shape. User input shape on left and query results on right.

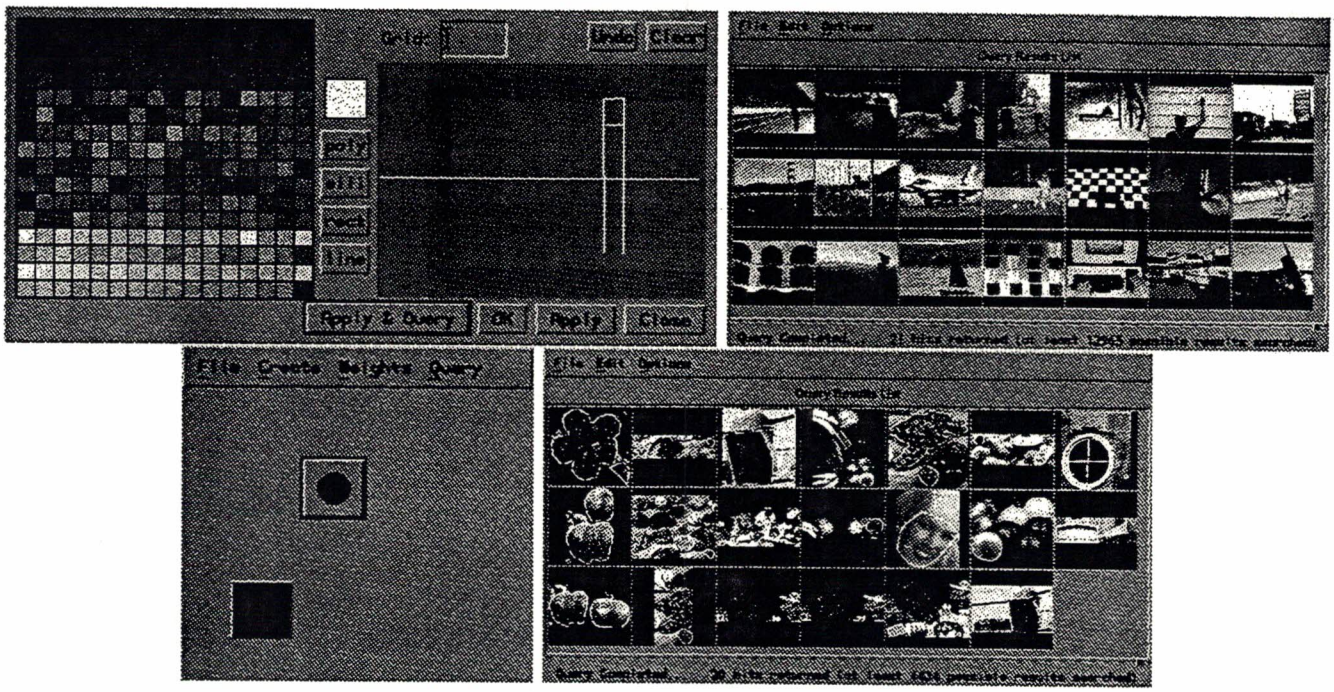


Figure 9: Top : Query by sketch. Sketched query specification on left. Top 21 hits from a 12965 image database shown on the right. Bottom: A Multi-* object query. The query specification on the left describes a query for images with a red round object and a green textured object. Top 20 matches shown on the right

5 Discussion and Future directions

Large online image and video collections are becomimg common – web sites contain thousands of images; digitial libraries are being generated by universities, museums, and government institutions; CD-ROMS contain thousands to tens of thousands of images; manufacturers and retail industries have image collections of over a million items. Better methods are needed to manage, organize, and retrieve data from these collections. We have described a prototype system that uses image and video content as the basis for retrievals. Technology from this prototype has already moved into a commercial product, IBM's Ultimedia Manager, and other companies are beginning to offer products with similar capabilities. Key challenges remain in making this technology pervasive and useful:

Annotation and database population tools. Automatic methods such as our Positional Color query that do not rely on object identification, methods that identify objects automatically as in the museum image example, fast and easy-to-use semi-automatic outlining tools, and motion based segmentation algorithms will enable additional application areas.

Feature extraction and matching methods. New mathematical representations of video, image, and object attributes that capture "interesting" features for retrieval are needed. Features that describe new image properties such as alternate texture measures or that are based on fractals or wavelet representations, for example, may offer advantages of representation, indexability and ease of similarity matching.

Integration with text and parametric annotation. Query by visual content complements and extends existing query methods. Systems must be able to integrate queries combining date, subject matter, price and availability with content properties such as color, texture, and shape.

Extensibility and flexibility. System architectures must support the addition of new features and new matching/similarity measures. Real applications often require new features, say a face matching module, to add to their existing content-based retrieval capabilities.

User interface. The user interface must be designed to allow easy selection of content-based properties, to allow them to be combined with each other and with text or parametric data, and to allow query reformulation and general database navigation.

Indexing and performance. As image and video collections grow, system performance must not slow down in proportion. Indexing, clustering, and filtering methods must be designed into the matching methods to maintain their performance.

With these technologies, the QBIC paradigm of visual content querying combined with traditional keyword and text querying will lead to powerful search engines for multimedia archives. Applications will occur in areas such as decision support for retail marketing, online stock photo and video management, cataloging for library and museum collections, and multimedia-enabled applications in art, fashion, advertising, medicine and science.

6 Sidebar: Semantic versus non-semantic information

At first glance, content based querying appears deceptively simple, because we humans seem to be so good at it. If a program can be written to extract semantically relevant text phrases from images, the problem may be solved using currently available text search technology. Unfortunately, in an unconstrained environment the task of writing this program is beyond the reach of current technology in image understanding. At an artificial intelligence conference a few years back, a challenge was issued to the audience "write

a program that identifies all the pictures of dogs given a children's book," a task most 3 year olds can easily accomplish. Nobody in the audience accepted the challenge and this remains an open problem. Perceptual organization, the process of grouping image features into meaningful objects, and attaching semantic descriptions through model matching to scenes are unsolved problems in image understanding. Humans are much better than computers in extracting semantic descriptions from pictures. Computers however, are better than humans at measuring properties and retaining these in long term memory. One of the guiding principles used by QBIC is to let computers do what they do best, quantifiable measurement, and let humans do what they do best, attaching semantic meaning. QBIC can find "fish shaped objects" since shape is a measurable property that can be extracted. However, since fish occur in many shapes the only fish that will be found will have a shape close to the drawn shape. This is not the same as the semantical query of finding all the pictures of fish in a pictorial database.

7 Sidebar: Fast searching and indexing

Indexing tabular data for exact matching or range searches in traditional databases is a well understood problem, and structures like B-trees provide efficient access mechanisms. In this scenario, indexing assures sublinear search while maintaining completeness, that is *all* records satisfying the query are returned without the need for examining each record in the database. However, in the context of similarity matching for visual content, traditional indexing methods may not be appropriate. For queries in which similarity is defined as a distance metric in high-dimensional feature spaces, for instance color histogram queries, indexing involves clustering and indexable representations of the clusters. In the case of queries that combine similarity matching with spatial constraints on objects the problem is more involved. Data structures for fast access of spatial relations of high-dimensional features need to be invented.

In QBIC, at query time, features from the database are compared against corresponding features from the query specification to determine which images are a good match. For a small sized database, sequential scanning of the features followed by straightforward similarity computations is adequate. But as the database size grows, this combination can be too slow. In order to speed up the queries, we have investigated a variety of techniques.

1. **Filtering.** A computationally fast filter is applied to all data, and only items that pass the filter are operated on by the second stage which computes the true similarity metric. For example, in QBIC we have shown that color histogram matching, which is based on a 256-dimensional color histogram and requires a 256 matrix-vector multiply, can be made efficient by filtering. The filtering step employs a much faster computation in a 3-dimensional space with no loss in accuracy. Thus, for a query on a database of 10,000 elements, the fast filter is applied to produce the best 1000 color histogram matches. These filtered histograms are subsequently passed to the slower complete matching to obtain, say the best 200 to display to a user, with the guarantee that the global best 200 in the database have been found.

2. **Indexing.** For low dimensional features such as average color and texture (each three dimensional), multidimensional indexing methods such as R^*-trees can be used. For high dimensional features – for example, our 20 dimensional moment-based shape feature vector – the dimensionality is reduced using the K-L or principal component transform. This produces a low dimensional space, as low as two or three dimensions, which could be indexed using R^* trees.

References

[1] Y. T. A. Akutsu, K. Otsuji, and T. Sadakata. VideoMAP and VideoSpaceIcon: Tools for anatomizing video content. In *ACM IN-*

TERCHI, pages 131–136, 1993.

[2] J. Ashley, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in QBIC. In *Image and Video Storage and Retrieval III*, volume 2420, San Jose, CA, February 1995. SPIE.

[3] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *Proc. Intl. Conf. on Computer Vision*, 1995.

[4] V. N. Gudivada and V. V. Raghavan. Picture retrieval systems: A unifed perspective and research issues. Submitted to ACM Computing Surveys, 1994.

[5] IFIP. *Visual Database Systems I and II.* Elsevier Science Publishers, North-Holland, 1989 and 1992.

[6] R. Jain, editor. *Workshop on Visual Information Management Systems*, Redwood, CA, February 1992. NSF.

[7] T. Kato, T. Kurita, and H. Shimogaki. Intelligent visual interaction with image database systems – toward the multimedia personal interface. *Journal of Information Processing (Japan)*, 14(2):134–143, 1991.

[8] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In *Visual Database Systems, II, IFIP Transactions A-7*, pages 113–127. Elsevier Science Publishers, 1992.

[9] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The QBIC project: Querying images by content using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187. SPIE, February 1993.

[10] W. Niblack and R. Jain, editors. *Storage and Retrieval for Image and Video Databases*, San Jose, CA, 1993, 1994 and 1995. SPIE. Volumes 1908, 2185 and 2420.

[11] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proceedings: Storage and Retrieval for Image and Video Databases II*, pages 34–47, San Jose, CA, February 7-8 1994. SPIE.

[12] H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2D&3D dominant motion estimation for mosaicing and video representation. In *Proc. Intl. Conf. on Computer Vision*, 1995.

[13] L. A. Teodosio and W. Bender. Salient video stills: Content and context preserved. In *ACM Intl. Conf. on Multimedia*, 1993.

[14] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 361–366, 1993.

PRO 0703
Writing Reusable Classes with C++

Noel Bergman
DevTech Technologies, Inc.
Columbia, S.C.

"Object-oriented programming has failed"

BYTE Magazine

"The rumors of my death have been greatly exaggerated"

Mark Twain

THE CHALLENGE OF REUSABLE CODE

Despite the tabloid-like quality of BYTE magazine's May 1994 headline, the issue's report on ComponentWare raises several interesting points. Jon Udell's assertion that object-oriented programming has failed, presents both a challenge and a springboard to practitioners of object-oriented programming. Despite claims that object-oriented programming provides substantial benefits in terms of code reuse, the empirical evidence is often to the contrary. Why is that, and what can be done about it? Is object-oriented programming really not the answer, and is ComponentWare some new paradigm to take us to the promised land of reusable code?

One answer is that ComponentWare is nothing new at all. It is, in many respects, a repackaging of tried and true object-oriented programming methods that date back more than 15 years. Those in the object-oriented community who are familiar with Smalltalk-80 will no doubt recognize echos of the Model/View/Controller (MVC) metaphor. Still others will recognize echos from other message based programming models.

Why then do we not see more benefits from reuse in today's market? Perhaps the answers lie in the means by which object-oriented solutions are delivered, both in terms of the most popular delivery vehicle, the C++ language, and the techniques employed by C++ programmers. Simply put, C++ does little to prevent one from making code reuse an almost virtual impossibility, and many C++ programmers lack the training necessary to deliver reusable code. Also inhibiting reuse is the lack of maturity within the C++ class library market, within which exists almost none of the standards, or even guidelines, that would help promote reusability. Very few C++ libraries are compatible with other C++ libraries, making reuse difficult. This is compounded by the seeming compulsion exhibited by many C++ programmers to develop their own class library, particularly when it comes to the ubiquitous types of data structures that seem to pervade so many class libraries. This rite of passage seems most prevalent amongst those C++ programmers with the most raw talent, but serves to further diminish overall reuse.

That said, it is fair to say that the apparent lack of reuse is only partially real; it is also largely matter of perception and scope. Many companies have successfully developed libraries of reusable C++ classes for internal use, and there are quite a few commercial class libraries. Most reuse in C++ seems to occur within the relatively small domain of a development group.

This contrasts greatly with the claims for reuse in ComponentWare. It is therefore prudent to consider the success of ComponentWare, and to develop concrete suggestions to help promote better code reuse in C++.

THE SUCCESS OF COMPONENTWARE

First off, what is ComponentWare, and why is it successful? Although there is more than one way to define an object model, the most typical models include the concepts of inheritance, polymorphism, and encapsulation. ComponentWare, as defined in the BYTE article, focuses on encapsulation. Everything that you need to know about a component is defined in its interface. One's mental energies are focused solely on how to interface to a stock, invariant, component.

This focus has significant benefits. The vast majority of programmers who work with components simply focus on stitching together existing components via whatever means they are capable. They may never need to learn how to develop a good, reusable component, or perhaps even have the talent to do so. They make reuse of as

much external code as possible their priority and chief delivery vehicle.

There is nothing in object-oriented philosophy (first and foremost, the notions of object-oriented programming form a *philosophy* - a system of principles for guidance) that excludes ComponentWare. Rather, ComponentWare is an emphasis on implementing certain of the design patterns found in object-oriented programming. And thus, ComponentWare is appropriately viewed as an example of an application of object-oriented principles, with a focus on standardizing the interfaces between components.
Another such application of object-oriented principles is the MVC metaphor, one of the classic design patterns found in object-oriented programming.. In this metaphor, there are three basic kinds of objects: Models, which encapsulate the behavior relevant to a concept, but lack any form of user interface; Views, which present information, but do not process it; and Controllers, which provide an interface for modifying information. Together, these three kinds of objects collaborate to bridge the gap between the information in the computer, and the user interface. A controller notifies a model to effect a user directed change; a model notifies all of its views when it has new or modified information to be presented. The interfaces and necessary behavior of these objects is standardized through inheritance, which also ensures polymorphic behavior. Indeed, one of the fundamental tenants of MVC is the notion of "pluggable views", where you can simply plug a new view into a model.

What is important about MVC, and makes it work, is that it represents a standard framework within which programmers develop their solutions. The MVC guidelines provide direction for programmers, and the standardized MVC interfaces both promote the reuse of existing code, and ensure that new code will seamlessly integrate with old code. Each element of the MVC triad can be considered a stand-alone, reusable component, to be reused over and over where the situation warrants.

There are many other examples which one could use to show the benefits of standardized interfaces to stand-alone components. Indeed, object-oriented programming is very much like other tools; having a full tool chest does you little good if you can't agree on interface standards.

INTERFACE STANDARDS: THE OMG

The foremost effort to define standards in the object-oriented programming community is the Object Management Group (OMG). Nearly every vendor of object-oriented technology participates with the OMG, with the exception of Microsoft. Likewise, many large companies that wish to reap the benefits of using object-oriented technology are members. The attraction is the OMG's goal: widespread, cross-industry deployment of inter-operable, multi-vendor, distributed object solutions to business problems.

To this end, they have defined what it means to be an object (the OMG object model), a mechanism to interface with objects (the Common Object Request Broker, which provides the means by which one invokes an operation on an object), and a set of core Object Services (standard interfaces to be supported by all OMG compliant systems). The principles behind the development of the OMG Object Services provide an excellent overview of the considerations necessary to deliver reusable code.

Many of the design principles behind the OMG Object Services were first put forth in a wonderful little white paper jointly submitted by HP and SunSoft (OMG document 92-2-10). In that paper, they discussed approaches to develop interfaces with long term viability and extensibility.

We will use the term "interface" to refer to type definitions, and the term "class" to refer to implementations of a type. Thus a derived interface will be referred to as a subinterface, and a derived class referred to as a subclass.

RECOGNIZE THE DIFFERENCE BETWEEN CLASS LIBRARIES AND FRAMEWORKS

Class libraries are collections of classes that may or may not relate to one another in any particular way, nor do the classes impose any great structure on how they are to be used. Frameworks, on the other hand, embody a

micro-architecture in some domain. They not only provide cohesive reusable code, but they provide patterns for how the code is to be reused. Good frameworks are much harder to develop, but deliver far greater value.

DEFINE INTERFACES, NOT IMPLEMENTATIONS

Our focus should be on the interface to the object, not on its implementation. Interfaces should be self-contained and minimalistic. The narrower the interface, the less opportunity for unnecessary notions to pollute the interface.

For the same interface, there can be multiple qualities of service. For example, the basic interface to a event service should not change regardless of whether or not it guarantees delivery, guarantees at-most-once delivery, deals with consumers synchronously or asynchronously, etc.. In fact, the object with which we are working might not be an event service implementation at all; it might be a wrapper object, which masquerades as a event service.

In C++, we might implement interfaces as well designed pure abstract base classes.

DO NOT BUILD NON-TYPE PROPERTIES INTO INTERFACES

The interface to an object defines its type. One should be careful not to pollute the interface with non-type related properties of the object. For example, although the "parent" of an object may be important to its implementation, it might not be an important part of its type. An example of this is the Dictionary class. Typically, a Dictionary is implemented as a subclass of a Set. In practice, this can lead to problems. The defined behavior of a Set is such that it rejects duplicate members. That is a quality that may or may not be desirable in a Dictionary. Were we to change this behavior for a Dictionary, and pass such a Dictionary through an interface expecting a Set, the consequences would be at best undefined.

For C++ programmers, this can be an excellent place for the use of protected or private inheritance. Dictionary can inherit from Set as an implementation matter, but remain separate from Set as a defined type.

Furthermore, the only state visible through the interface should be abstract state at the same level of discourse as the interface. Implementation details should not be accidentally exposed through the interface showing implementation dependent state elements.

USE SUBINTERFACES TO EXTEND, EVOLVE, AND SPECIALIZE FUNCTIONALITY

Just as we typically think to extend an implementation by deriving a subclass, when extending an interface, we should do so by deriving a specialized subinterface. In this manner, code that is dependent upon the original interface can continue to function, while permitting the interface to evolve and extend in new directions.

In the case of an event service, a subinterface might permit controlling such factors as delivery policy, filtering, etc.. Those facilities would be available to code written to take advantage of the subinterface, but all event service users would continue to be able to use the original interface.

USE MULTIPLE INHERITANCE

Each type is defined as an interface. Each interface should be kept to the minimum necessary to specify the type. Rather than have large monolithic interfaces, we should have a set of narrow interfaces that can be combined to provide the same overall effect as the monolithic interface, but can also be combined in myriad other ways. This results in far more flexible and extensible schema; schema more tolerant of evolutionary change than those formed from monolithic interfaces.

Multiple inheritance is particularly important in C++ because of the C++ object model. Runtime polymorphism in C++ requires that the expected type and real type share compatible virtual functions. Classes which

exhibit such compatibility are said to belong to the same polymorphic cluster.

MULTI-OBJECTS

Sometimes composing an interface through multiple inheritance isn't the best approach. Some interfaces are better specified as being made up of discrete objects that collaborate synergistically to provide a gestalt. The OMG Event Notification Service provides an example of this kind of interface.

Rather than define an event service as a single interface that supports several discrete notions, such as supplier, consumer, and event management, it is more appropriate to view the event service as providing the means to acquire objects that in turn provide each of the component interfaces. Thus, rather than perform supplier administration directly with the event service, we use the event service to acquire an object whose interface provides for supplier administration.

In C++, this is often another excellent place to use protected or private inheritance. The reason for this is that we can gain all of the advantages of inheritance, from an implementation perspective, while preserving the pure interface seen by client code.

REUSABLE C++

To summarize some of the things thus far suggested for C++ programmers:

- Implement interfaces as pure abstract base classes. This allows the creation of what are called *polymorphic clusters* or *homomorphic type hierarchies*. Interfaces are often implemented as virtual base classes.
- Carefully consider inheritance from a type perspective. Use protected and private inheritance, as necessary, to keep interfaces true to type, while still gaining the benefits of inheritance.
- Keep each interface to a minimum, and use multiple inheritance to compose several separable interfaces into a single interface when necessary.
- Where appropriate, design wider interfaces as multi-objects. Again, protected or private inheritance can be used to keep the advantages of inheritance, from an implementation perspective, while preserving the pure interface seen by client code.

These are all generally applicable techniques, with only some C++ specificity involved. There are some other techniques in C++ that are underused, but can result in code that is either more reusable, or at least doesn't interfere with the usability of other code.

INDIRECTION AND DELEGATION

In the common course of events, we think of one object as invoking an operation on another object. C++ provides rich facilities that enable us interpose another object between the two primary objects, without either one being any the wiser. The most commonly used indirection mechanism in C++ is the smart pointer; an object with an overloaded operator->. Through this, or other C++ facilities, the invoking object perceives a pointer or reference to the desired target type. The interposed object determines how and when the target object is made available to the calling object, transparently managing availability and accessibility on the programmer's behalf. Ambitious use of indirection techniques can even provide the means to implement an easy to use UNDO framework.

NESTED TYPES

Pollution of the global name space is one of the most significant obstacles to federating multiple C++ class libraries. This is recognized and addressed in the ANSI C++ draft by the new namespace mechanism. In the meantime, C++ programmers can employ the underused nested type syntax to organize and isolate names logically.

Some uses of nested types:

- Types that might otherwise be global can be made members of a wrapper class. For example, if there were some need to define a type called Boolean, it could be referenced as <LibName>::Boolean, rather than left in global scope.
- Functions and variables that might otherwise be global can be made static members of a wrapper class.
- Types that are subordinate and intimately associated with another type can be nested within the superior type. For example, the general interface for a container might define the general interface for an iterator. In this way, if there is another general interface defined for a container, perhaps from some other library, the two different iterator definitions will not collide, and will remain associated with the correct container type.

Again, the goal is to avoid the kind of name space collisions that so frequently prevent the coexistence of multiple C++ libraries.

ACTORS

The term *actor* has overloading meaning within the object-oriented programming community. Here we use it to refer to what Coplien calls *functors;* objects that act like functions. The actor provides a standard interface able to redirect the invocation to any of many different implementations. The actor interface should define the parameter and return types, and operator(). To preserve strong typing, and assist in the development of strongly type polymorphic clusters, the actor hierarchy should likely be implemented using the C++ template syntax. Once defined, actors often prove to be amongst the most prolifically reused interfaces.

MESSAGE BASED PROGRAMMING

Message based programming is nothing new, but like object-oriented programming, the last few years have seen a tremendous surge in its proliferation. One reason for the growth of message based programming is that it can lead to massive reuse, and extensibility. Messaging interfaces are often simple and remain largely fixed; it is the message content that needs to be extensible. Message based programming is ideal for scalable, transparently distributed systems, and provides almost impossible to beat support for software evolution.

For our purposes, we can think of message based programming as a metaphor in which objects exist and exchange messages. The interface to such objects concerns the exchange of messages, rather than the detailed behavior of the objects themselves. We can view these objects as components residing on a software bus, which serves to move messages between components.

This approach to programming is a primary implementation vehicle underlying distributed programming platforms. The OMG's CORBA architecture is often implemented this way. The objects with which one exchanges messages are those which manage the actual objects upon which one desires to invoke an operation. The message exchanged identifies the actual target object, the operation to be performed, and any necessary parameters.

Microsoft's OLE is another such message based architecture. The actual implementation of the OLE objects isn't

important. What is important is OLE's messaging interface, and the format of the OLE messages. Other examples of message based systems include IBM's Message Queue Interface (MQUEUE), Apple's AppleEvents, SNMP, CenterLine's ObjectCenter, and HP's SoftBench.

Perhaps more important, in today's market, is the fact that message based systems are easily language and platform independent. It is often a relatively straightforward matter to gateway between disparate message systems. A single message based component can be accessed from message systems other than the one on which it is hosted. Thus, a server component can provide services to clients of many types, not just those originally intended by the server's author.

Using the techniques discussed earlier in this paper, C++ can be used to easily develop high performance, message based systems capable of long term stability.

EMERGING TECHNOLOGIES

Over the next few years, there will be a lot of change in software development. The widespread deployment of CORBA compliant object systems will promote the development of guidelines and standards that will greatly enable interoperability of objects, and further the goals of reuse. IBM, MetaWare, and others have C++ compilers that natively generate SOM objects. SOM (System Object Model) is the technology underlying IBM's own CORBA implementation, as well as much of the object technology developed by IBM and partners such as Component Integration Laboratories (the OpenDoc consortium), Apple, Novell, WordPerfect, Borland, Lotus, HP, and others.

The emergence of frameworks such as Commonpoint, and others will also tend to gather compatible class libraries in their sphere of influence, particularly those frameworks that are backed by language, platform, and vendor independent technology, such as OpenDoc and Taligent's Commonpoint which are underpinned by SOM.

Perhaps the most important change in software development will be the widespread recognition of the concept that good architectures and designs contain a relatively small set of design patterns, albiet implemented in different ways. The recognition of the existence of these design patterns, and their widespread use, will lead to more understandable and reusable code.

FURTHER STUDY

Here are are a few references for further study and deliberation. The list is intended to provide pointers to different approaches and views, rather than ennumerate a lengthy collection of sources that few will take the time to read.

Design Patterns: Elements of Reusable Object-Oriented Software; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; Addison-Wesley; ISBN 0-201-63361-2. This book is an excellent introduction to the notion that good software architecture, like good architecture in other fields, is based upon the intelligent use of design patterns.

Advanced C++ Programming Styles and Idioms, James Coplien, Addison-Wesley, ISBN 0-201-54848-0. Mr. Coplien illustrates a number of techniques for implementing large, robust, extensible systems in C++.

Plug & Play Programming, William Wong, M&T Press, ISBN 1-55851-302-7. Mr Wong presents a homegrown C++ framework for the development of message based programs using the notion of plugs and sockets.

Frameworks: The Journal for Software Developers Using Object Technology, published by the Software Frameworks

Association (408-253-2765). Ran a series of articles on C++ for Gurus in early 1994 by Jeff Alger that were derived from his "Advanced C++" seminar taught at Apple's Developer University.

The Object Management Group. The OMG has an extensive collection of documents, ranging from white papers to descriptions of OMG endorsed technologies. The aforementioned white paper (OMG document 92-2-10) is only a single instance of the excellent reading material to be had by anonymous FTP. To receive further information, write a message to omg_server@omg.org with the word "help" in the message body.

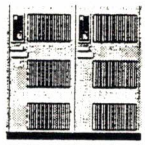
PRO 0706
IBM COBOL Goes Desktop!

Phil Arthur
IBM

The COBOL Environment Today

IBM®

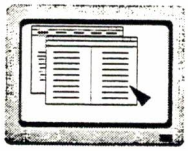
Application Development Solutions



- Investment
- Maintenance

- Skills Assessment
- Viability
- Retraining

- Cost
- Training
- Productivity



- Improved Productivity
- Greater Risks

- End-User needs

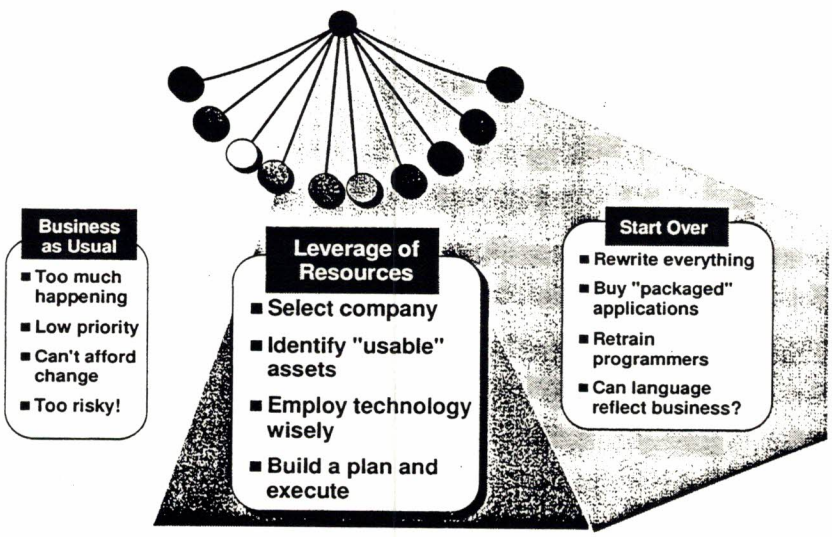
Application Development Team

© COPYRIGHT IBM CORPORATION 1995

Spectrum of Options

IBM®

Application Development Solutions



© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Strategy

IBM®

Application Development Solutions

- Workstation Development
- Language Consistency with Optimized Performance
- Technology Enhancements
- Integrated Maintenance and Testing
- Team Development
- Solutions to Build Your Plan

© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Family Overview

IBM®

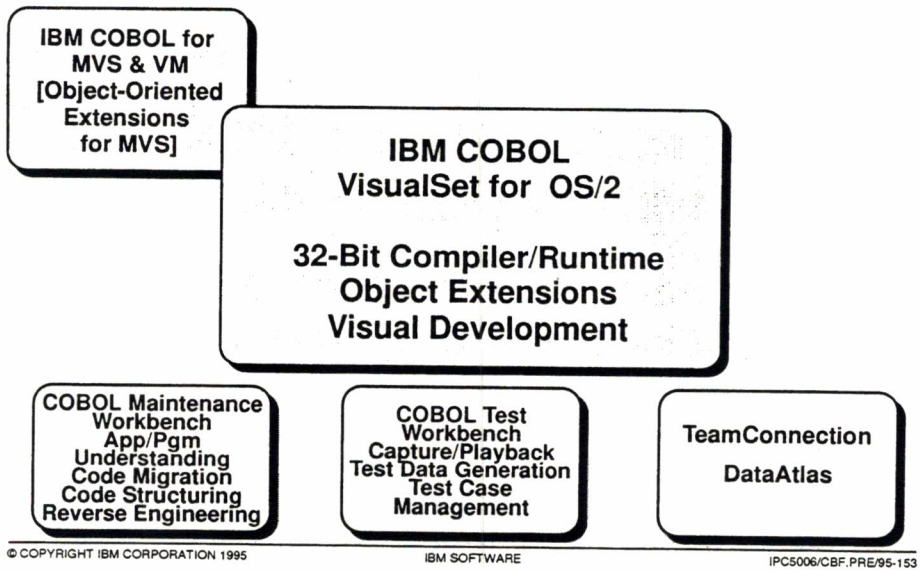
Application Development Solutions

© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Family for OS/2 Overview

IBM®

Application Development Solutions



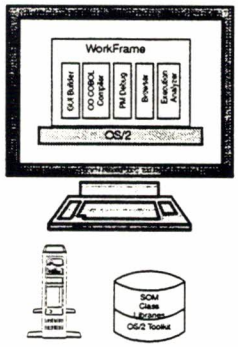
© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Solution

IBM®

Application Development Solutions

- IBM COBOL VisualSet for OS/2

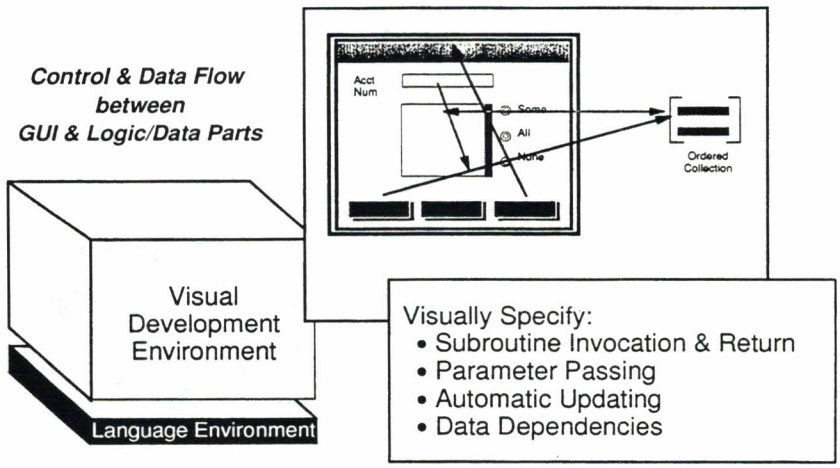


- **IBM COBOL Compiler**
  - ✓ Robust,Optimizing
  - ✓ Standards Conforming
  - ✓ ANSI '85 COBOL
  - ✓ ANSI OO COBOL (Draft)
  - ✓ "Direct-to-SOM"compiler (for OO)
  - ✓ Based on COBOL for MVS & VM
- **Graphical Debugger**
- **Visual GUI Builder**
  - ✓ Visual creation
  - ✓ Event Driven GUI
  - ✓ Code Assistant
  - ✓ Generate Logic in COBOL
- **Enhanced Editor**
  - ✓ Data assistant for DB2 access
  - ✓ Transaction assistant for CICS ECI calls
  - ✓ Live parsing
  - ✓ Configurable use of color
- **Execution Analyzer**
  - ✓ Visual Optimization and Tuning of Complex Applications
- **Developer's Toolkit**
  - ✓ Linker and Make Utilities
  - ✓ Presentation Manager Tools
- **WorkFrame**
  - ✓ Full Workplace Shell Integration
  - ✓ Project Organizer
  - ✓ Open and Configurable Tool Integration Framework
- **Access to data:**
  - ✓ DB2 local & remote
  - ✓ IMS remote access
  - ✓ VSAM local & remote
- **SOM Class Libraries**
  - ✓ Base Set of Classes
  - ✓ Collection Classes

© COPYRIGHT IBM CORPORATION 1995

Aspects of Visual Development

Application Development Solutions

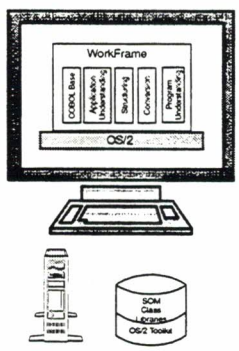


© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Solution

Application Development Solutions

- IBM COBOL Maintenance Workbench for OS/2



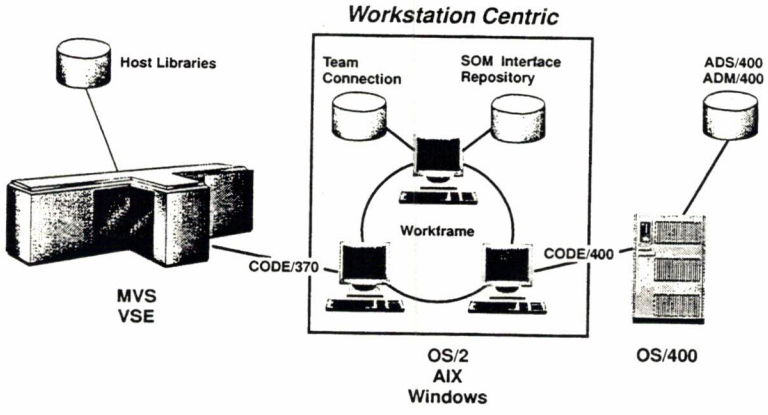
- **Application Understanding**
  - ✓ JCL
  - ✓ DD statements
  - ✓ Executable (load module)
- **Program Understanding**
  - ✓ System structure
  - ✓ Subsystem structure
  - ✓ Internal program units
  - ✓ Impact analysis
  - ✓ Graphical display
    - calling strcuture
    - control flow
    - data structures
- **Structuring**
  - ✓ Converts unstructured COBOL to structured COBOL
  - ✓ Improves maintainability
  - ✓ COBOL/SF engine on OS/2
- **Conversion**
  - ✓ Automatic conversion of ANSI 68 and 74 COBOL source to ANSI 85 standard
  - ✓ Converts source program & copy modules
- **Metrics**
- **Slicing**
- **Requires IBM COBOL VIsualSet for OS/2**
- **Integrated via WorkFrame/2**

© COPYRIGHT IBM CORPORATION 1995

Development Platforms

IBM®

Application Development Solutions



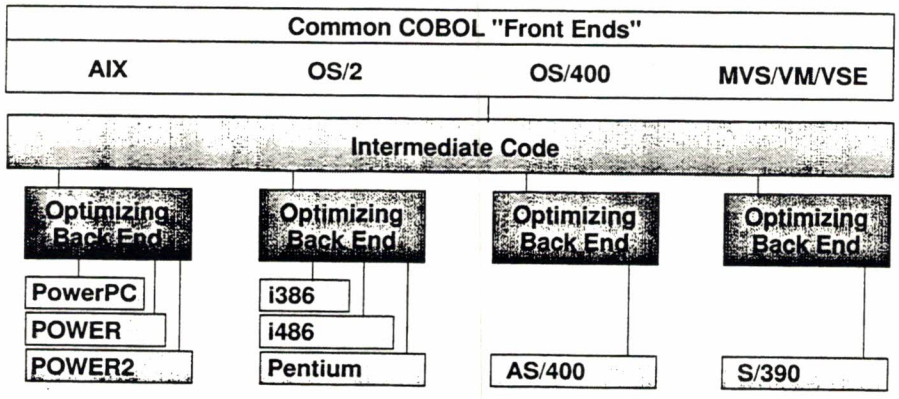
© COPYRIGHT IBM CORPORATION 1995

COBOL Compiler Technology

IBM®

Application Development Solutions

- Advanced Optimizing Compiler Architecture

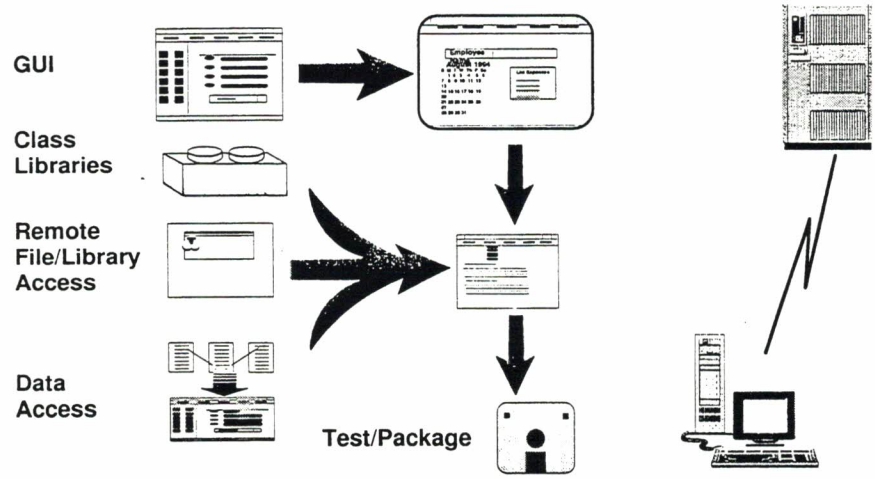


© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Family: Client/Server Application Scenario

IBM®

Application Development Solutions

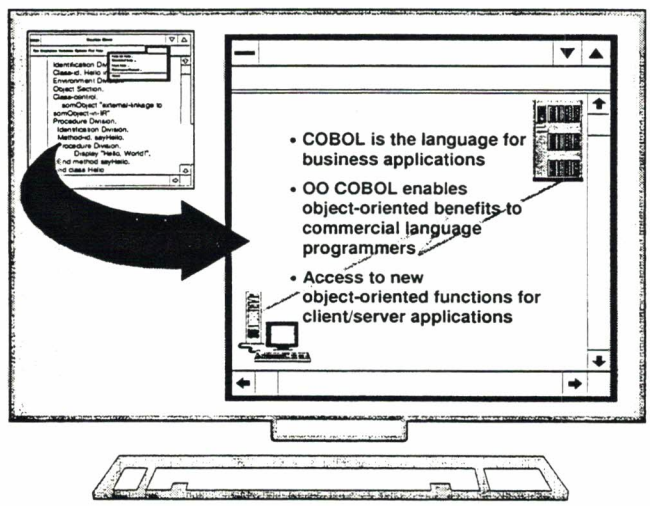


© COPYRIGHT IBM CORPORATION 1995

Why Object-Oriented COBOL?

IBM®

Application Development Solutions



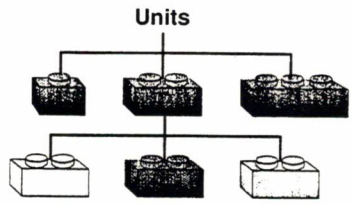
© COPYRIGHT IBM CORPORATION 1995

IBM Object COBOL: Overview

IBM®

Application Development Solutions

- Enables System Object Model (SOM), via new COBOL Syntax
  - Class definition syntax
  - INVOKE verb
- Direct-to-SOM Compiler
- SOM Interface Repository used at Compile-Time
- Access to SOM-Enabled class libraries

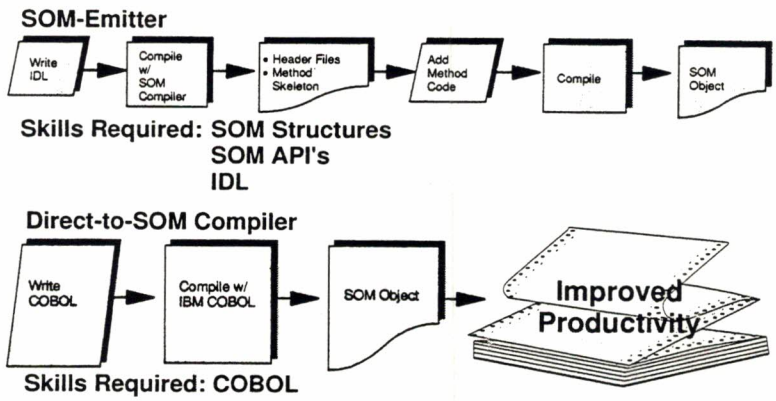


© COPYRIGHT IBM CORPORATION 1995

Direct-to-SOM Compiler vs. SOM-Emitter

IBM®

Application Development Solutions

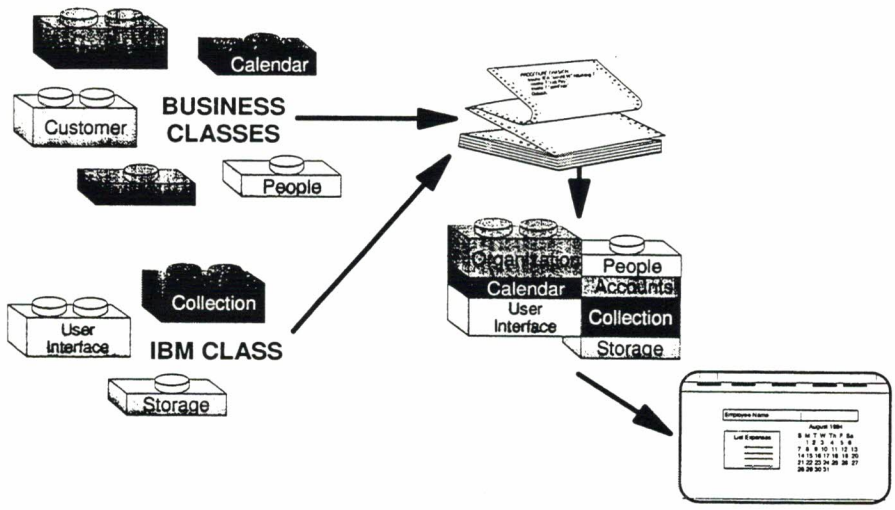


© COPYRIGHT IBM CORPORATION 1995

Object-Oriented COBOL: Construction from Parts

IBM®

Application Development Solutions

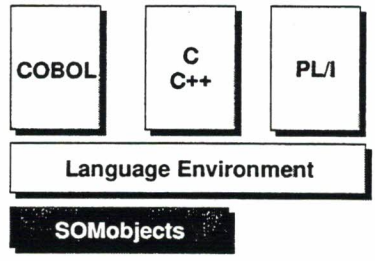


© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Language Interoperability

IBM®

Application Development Solutions



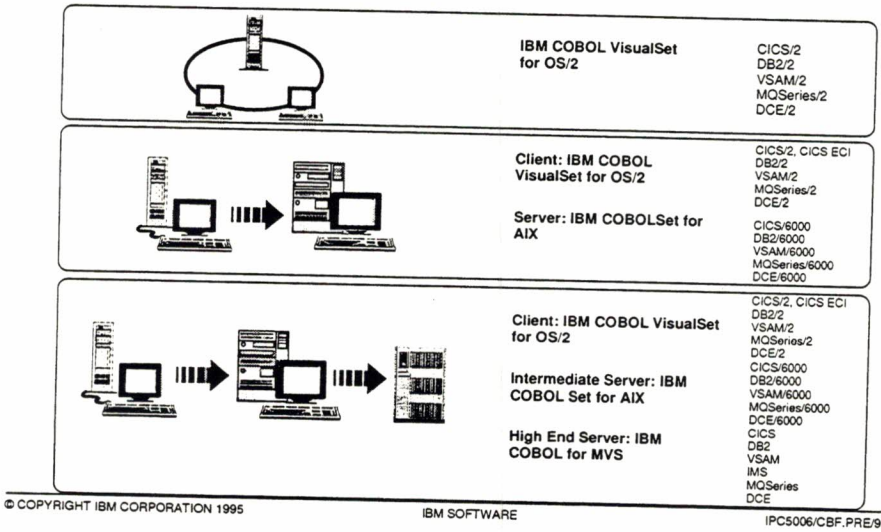
- Language Environment provides ILC on S/390
- SOM support provides "language neutral" classes
- Improved COBOL & C/C++ interoperability

© COPYRIGHT IBM CORPORATION 1995

Using the IBM COBOL Family for Client/Server

IBM®

Application Development Solutions

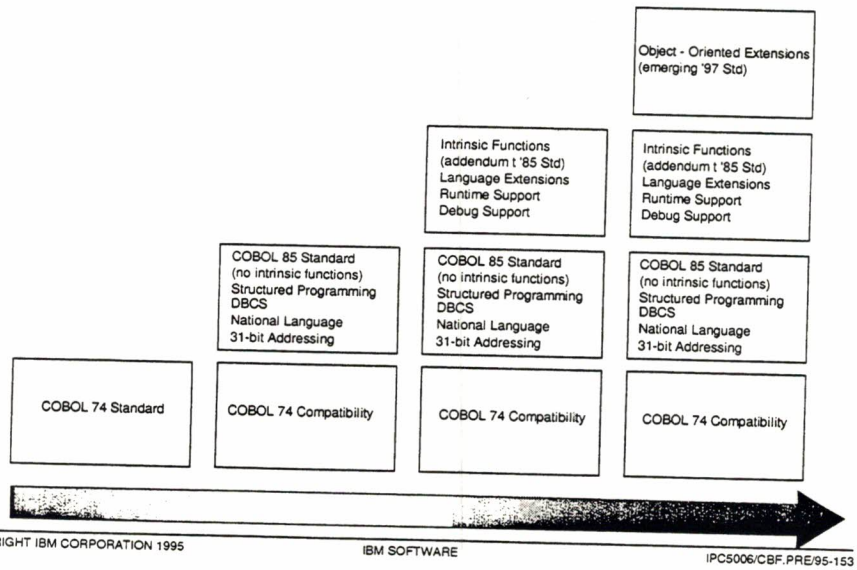


© COPYRIGHT IBM CORPORATION 1995

Evolution of IBM COBOL

IBM®

Application Development Solutions

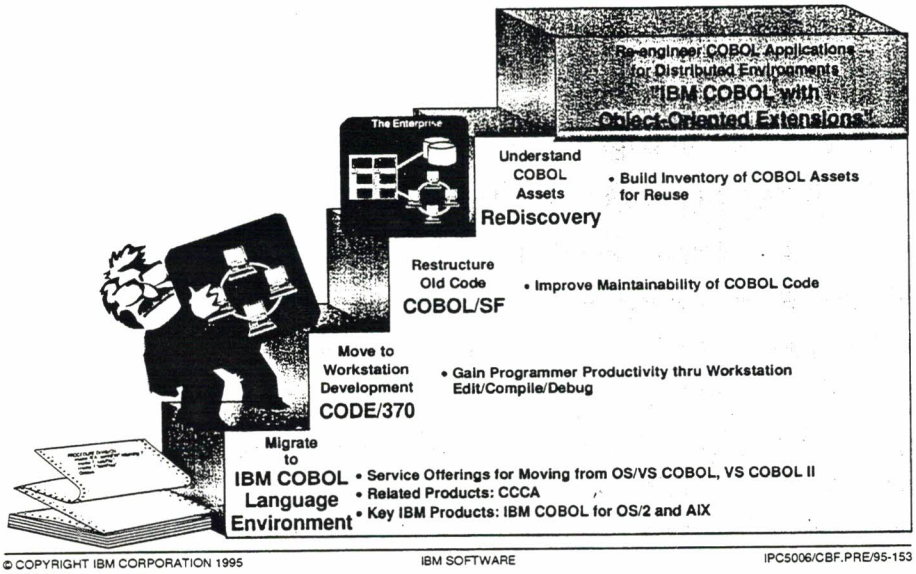


© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Family Client/Server Roadmap

IBM®

Application Development Solutions



© COPYRIGHT IBM CORPORATION 1995

IBM AD Principles:

IBM®

Application Development Solutions

- Integration and leverage of all design skills
- Iterative, team-oriented development
- Visual construction from parts
- Wide choice of languages and platforms
- Evolutionary transistion to objects

© COPYRIGHT IBM CORPORATION 1995

IBM COBOL Benefits

Application Development Solutions

- Programmer Productivity
- Investment Protection
  - Programmer Skills
  - Existing Applications
- Client/Server Development
- Procedural Code enhanced with Object-Oriented Extensions
- Compiler and Source Compatibility across Multiple Platforms

© COPYRIGHT IBM CORPORATION 1995

PRO 0707

Migrating from Graphical Applications to Object-Oriented User Interfaces

Theo Mandel

Interface Design and Development
Austin, Texas

Theo Mandel, Ph.D.

- Founder and CEO:
  Interface Design and Development (IDD)
- Author:
  The GUI-OOUI War: Windows vs. OS/2, The Designer's Guide to Human-Computer Interfaces (John Wiley & Sons, ISBN: 0-471-13149-0)
- Consulting and Education Services:
  Graphical and Object-Oriented
  User Interface Design and Development
- Work with vendors and Fortune 500 companies
- Member of the CUA Interface Architecture group responsible for IBM and Microsoft user interface design guidelines and standards

1

Copyright 1995 Theo Mandel, Ph.D.

For Further Information, Please Contact:

Theo Mandel, Ph.D.
Interface Design and Development
8001 Two Coves Drive
Austin, Texas 78730-3125

(512) 345-2259 Phone/Fax

CompuServe: 75143,2160
Internet: 75143.2160@compuserve.com

2

Copyright 1995 Theo Mandel, Ph.D.

What to Expect from this Presentation

- WHAT are the differences between Application-Oriented user interfaces and Object-Oriented user interfaces
- WHY you need to focus on migrating your design and development efforts from Applications to Objects
- HOW an Object-Oriented and User-Centric design and development process works

Awareness | Understanding | Excitement

3

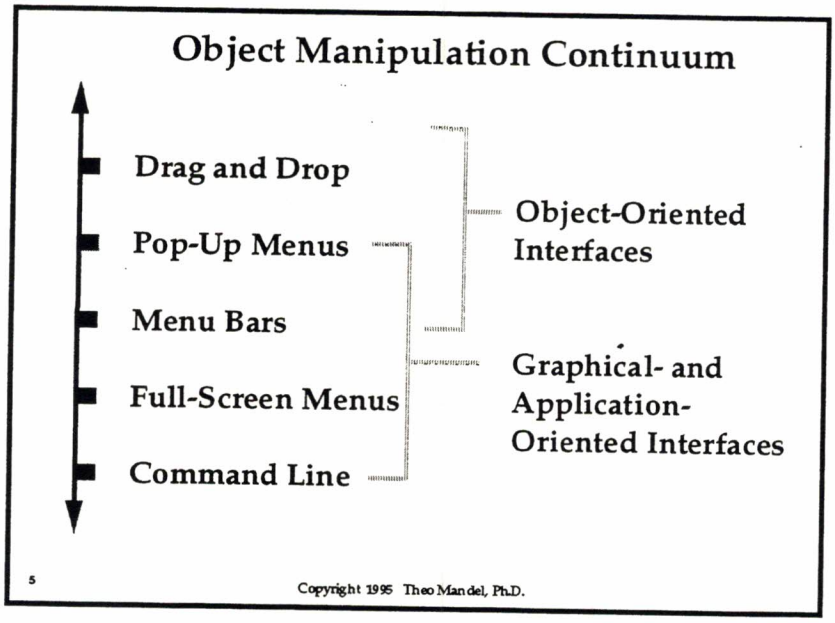
Copyright 1995 Theo Mandel, Ph.D.

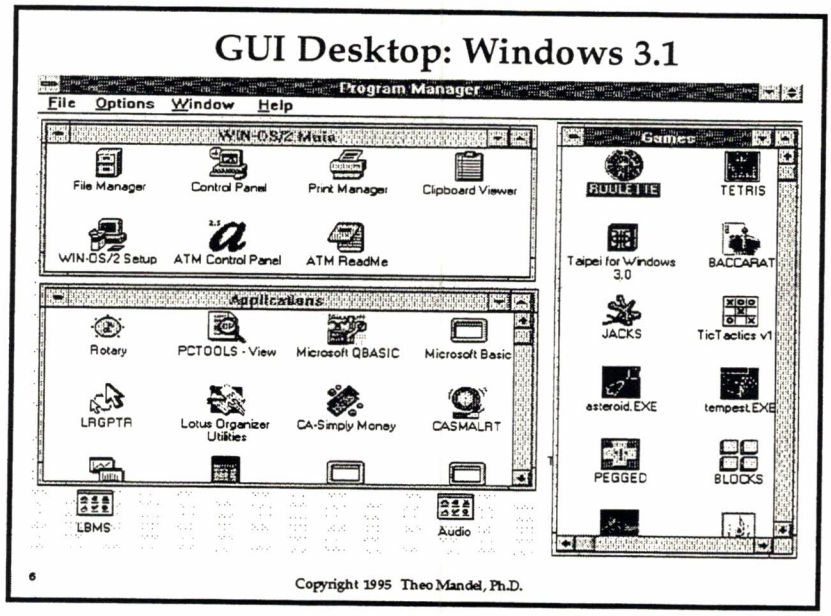
Key Differences between GUIs and OOUIs

GUI Interfaces	OOUI Interfaces
■ Applications	■ Product is a set of related objects
■ Data files	■ User data are objects, not data files
■ Application functions	■ User does tasks, not application functions
■ User is required to have some knowledge of the system (drives, paths, file names, etc.)	■ User should not need to know system information

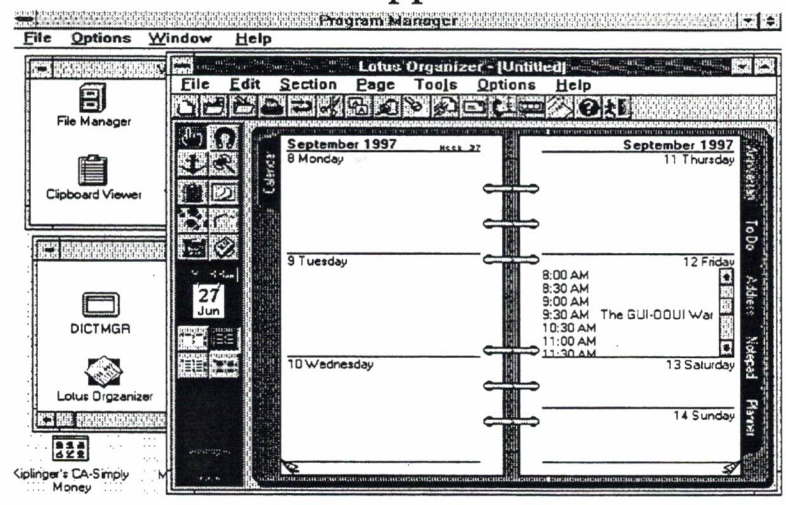
4

Copyright 1995 Theo Mandel, Ph.D.


Object Manipulation Continuum
Drag and Drop
Pop-Up Menus
Menu Bars
Full-Screen Menus
Command Line
Object-Oriented Interfaces
Graphical- and Application-Oriented Interfaces
5
Copyright 1995 Theo Mandel, Ph.D.


GUI Desktop: Windows 3.1
Program Manager
File Options Window Help
WIN-OS/2 Main
File Manager
Control Panel
Print Manager
Clipboard Viewer
WIN-OS/2 Setup
ATM Control Panel
ATM ReadMe
Applications
Rotary
PCTOOLS - View
Microsoft QBASIC
Microsoft Basic
LRGPTR
Lotus Organizer Utilities
CA-Simply Money
CASMALRT
LBMS
Audio
Games
ROULETTE
TETRIS
Taipei for Windows 3.0
BACCARAT
JACKS
TicTactics v1
asteroid.EXE
tempest.EXE
PEGGED
BLOCKS
6
Copyright 1995 Theo Mandel, Ph.D.

GUI Application



7

Copyright 1995 Theo Mandel, Ph.D.

Microsoft's GUI-OOUI Evolution

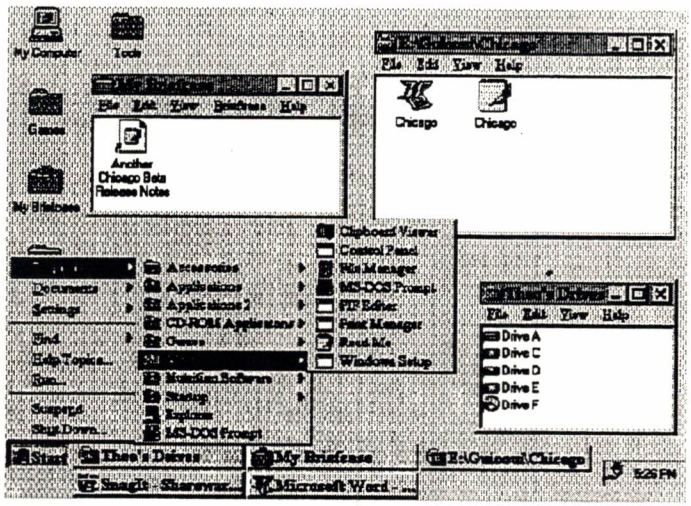
- Microsoft Windows User Interface Design Guide
  Preliminary Draft 4.00, May 1994

"The enhancements in the Windows interface pave the way for the evolution from basic graphical user interface design to a more object-oriented user interface design; that is, a more data-centric, rather than application-centric, interface...This is particularly important since this means from the user's perspective that applications become the engines behind the objects in the interface, enabling user interaction, but as transparent to users' tasks as the code that manages menus. This means that while applications will still be important, they no longer are the primary focus of the user. Users should be able to interact with their data without having to think about starting an application, allowing them to better concentrate on their tasks."

8

Copyright 1995 Theo Mandel, Ph.D.

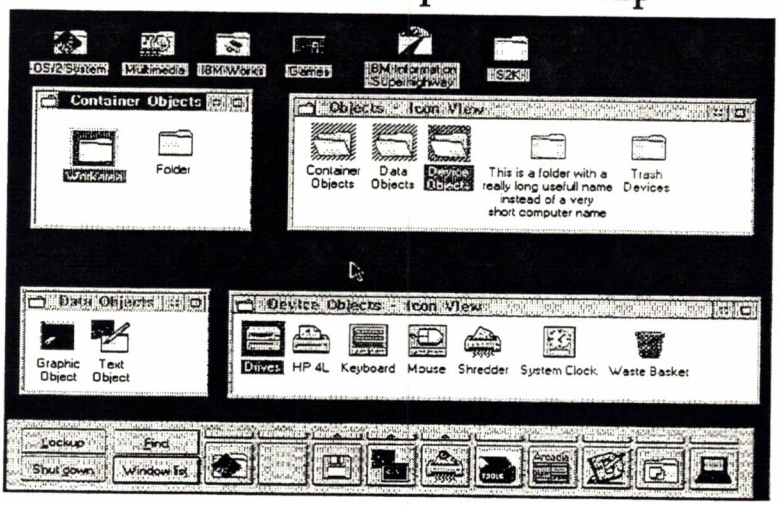
GUI-OOUI Desktop: Windows 95



9

Copyright 1995 Theo Mandel, Ph.D.

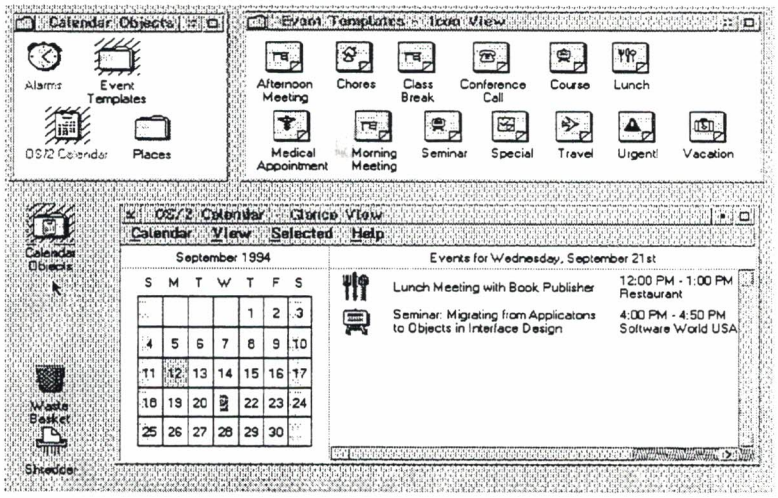
OOUI Desktop: OS/2 Warp



10

Copyright 1995 Theo Mandel, Ph.D.

Object-Oriented Product



11

Copyright 1995 Theo Mandel, Ph.D.

OOUIs: Views of Objects

- FIRST: Take great care to determine WHAT objects users need
- SECOND: You must also determine HOW users need (and want) to work with the objects
  - What are the appropriate views?
  - What users need what views?
  - What views are used when?
  - What media should be used?
  - What business process and security measures affect objects and views?
  - Views should be user and context sensitive

12

Copyright 1995 Theo Mandel, Ph.D.

Why Migrate From Applications to Objects?

- Users perform their own particular TASKS, not computer FUNCTIONS
- Users work with OBJECTS, not APPLICATIONS
- Users should be provided with their information When they want it and How they want it
- Users know what they want to do and how they want to do it, applications don't
- Objects follow REAL-WORLD METAPHORS, applications don't necessarily
- Operating Systems are becoming more OBJECT-ORIENTED, so products need to follow to remain state-of-the-art

3

Copyright 1995 Theo Mandel, Ph.D.

OOUIs and OOP

- OOP = Object-Oriented Programming
- OOUIs and OOP are assumed to always coexist:
  - If you have one then you have the other
  - NOT TRUE!
- Interface OBJECTS don't necessarily parallel programming OBJECTS
- OOUIs don't (formally) require Object-Oriented Programming (but it helps!)
- OOP doesn't automatically provide usable OOUIs
- OOUIs and OOP complement each other, but they are very different!

14

Copyright 1995 Theo Mandel, Ph.D.

An Object-Oriented User Interface Design Process

idd

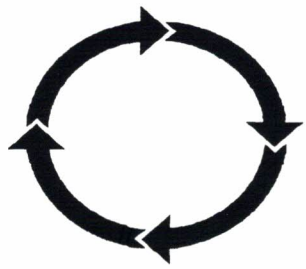
15

Copyright 1995 Theo Mandel, Ph.D.

An Iterative Design Process

Test

Plan



Proto-
type,Build

Design

16

Copyright 1995 Theo Mandel, Ph.D.

An Iterative Design Process

1. Gather Requirements and Plan
2. Design the Interface
3. Prototype and Build the Interface
4. Test the Interface

Repeat these steps iteratively...

17

Copyright 1995 Theo Mandel, Ph.D.

Phase 1: Gather Requirements and Plan

1. Create USER profiles
2. Analyze USER tasks
3. Evaluate USER work environments
4. Gather USER requirements

18

Copyright 1995 Theo Mandel, Ph.D.

Users and Tasks

Users	Tasks
■ Salesperson	■ Work with customers ■ Sell cars ■ Complete worksheets
■ Manager	■ Approve sales ■ Manage salespeople ■ Order new cars

19

Copyright 1995 Theo Mandel, Ph.D.

What You Need to Know about Users

- Cognitive and perceptual characteristics
- User motivation and expectations
- Prior knowledge and experience
- Job responsibilities and tasks
- Physical characteristics
- Social and physical environment
- Cultural environment, customs, and traditions

HOW do you learn these things?

- User task analysis
- User surveys, interviews, feedback
- Usability testing

20

Copyright 1995 Theo Mandel, Ph.D.

Phase 2: Design the Interface

1. Develop scenarios
2. Define interface objects and actions
3. Determine visual representations
4. Design menus
5. Refine visual designs

21

Copyright 1995 Theo Mandel, Ph.D.

Typical Scenario

- Customer asks for a particular car
- Salesperson looks up car in car lot listing
- Customer wants to see available colors & options
- Customer wants to see purchase finance options
- Salesperson collects customer information
- Customer decides to purchase car
- Customer wants to know value of trade-in car
- Salesperson completes order worksheet
- Salesperson sends order to manager for approval

22

Copyright 1995 Theo Mandel, Ph.D.

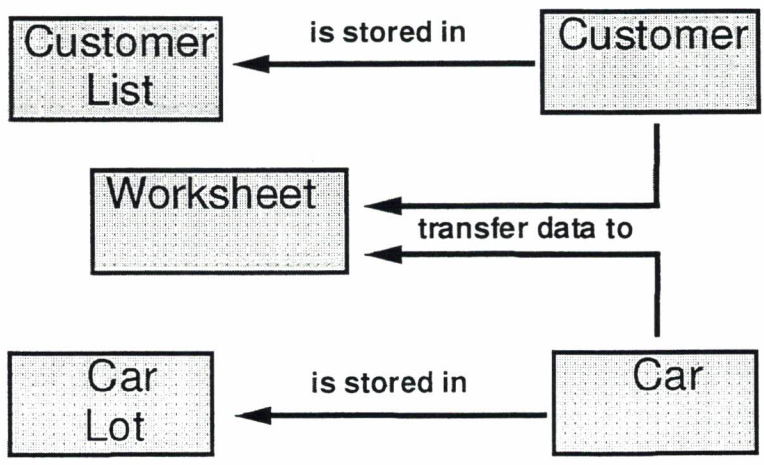
Object and Action Lists

Objects	Actions
■ Customer	■ Buy
■ Car	■ Sell
■ Salesperson	■ Transfer Data
■ Worksheet	■ Approve
■ Customer List	■ Move
■ Car Lot	■ Order

23

Copyright 1995 Theo Mandel, Ph.D.

Object Relationships Diagram



24

Copyright 1995 Theo Mandel, Ph.D.

Direct Manipulation Matrix

Target Object

Source Object	Car	Customer	Worksheet	Cust. List	Car Lot		
Car			Data transf.		Car moved		
Customer			Data transf.	Cust. moved			
Worksheet							
Cust. List							
Car Lot							

25

Copyright 1995 Theo Mandel, Ph.D.

Menu Layout Charts

Object Name: Customer View Name: Information

Menu Bar Item: Customer
Pull-down Choices:

Open as Information
Print
Send...

Menu Bar Item: Edit
Pull-down Choices:

Undo
Redo
Cut
Copy
Paste
Clear
Find
Select all
Deselect all

26

Copyright 1995 Theo Mandel, Ph.D.

Phase 3: Prototype and Build

1. Prototype early and iteratively
2. Prototype different design alternatives
3. Types of prototypes
   - Paper and pencil (flipcharts, storyboards)
   - Animated prototypes
   - Functional prototypes
4. Use prototypes as
   - Product and interface functional specs
   - Marketing and sales tool
     (to show managers and customers)

27 Copyright 1995 Theo Mandel, Ph.D.

Phase 3: Prototype and Build

5. Choose prototyping tools carefully

   - Prototyping tool:
     - *Typically produces throw-away code*
     - *Limited in functionality*
     - *Easy to use for non-programmers!*

   - Development tool:
     - *Typically produces code that may be used in the product*
     - *Rich in functionality*
     - *Difficult to use for non-programmers!*

28 Copyright 1995 Theo Mandel, Ph.D.

Phase 4: Test the Interface

Iterative Testing

1. Informally test early prototypes
2. Iterative formal usability testing
   - Conceptual prototypes
   - Separate parts of the product
3. Final usability and integrated product test
   - Icons, views, windows, menus...
4. Post-ship field testing and evaluations
5. Learn from design and development efforts
   - Perform post-mortem reviews

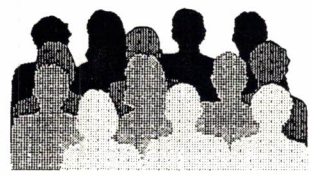
29

Copyright 1995 Theo Mandel, Ph.D.

Design Team Approach

Design Team Skills Required

- Programming
- User Interface Design
- Instructional Design
- Graphic Design
- Technical Writing
- Usability Testing
- Team includes:
  - Business Knowledge Experts
  - Users



30

Copyright 1995 Theo Mandel, Ph.D.

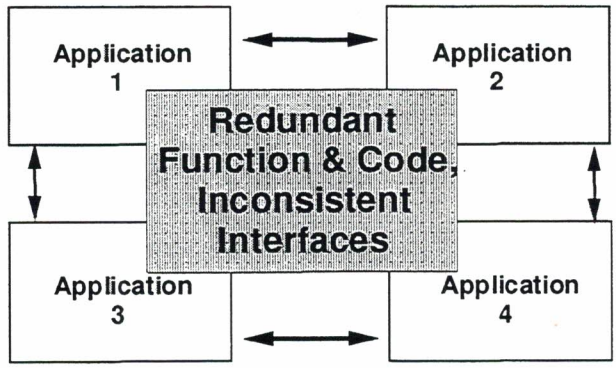
Where Applications are Going

idd

31

Copyright 1995 Theo Mandel, Ph.D.

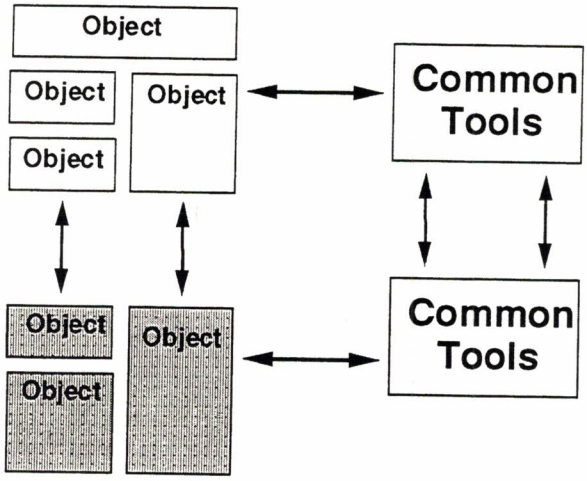
Today's Applications



32

Copyright 1995 Theo Mandel, Ph.D.

Tomorrow's Products



33

Copyright 1995 Theo Mandel, Ph.D.

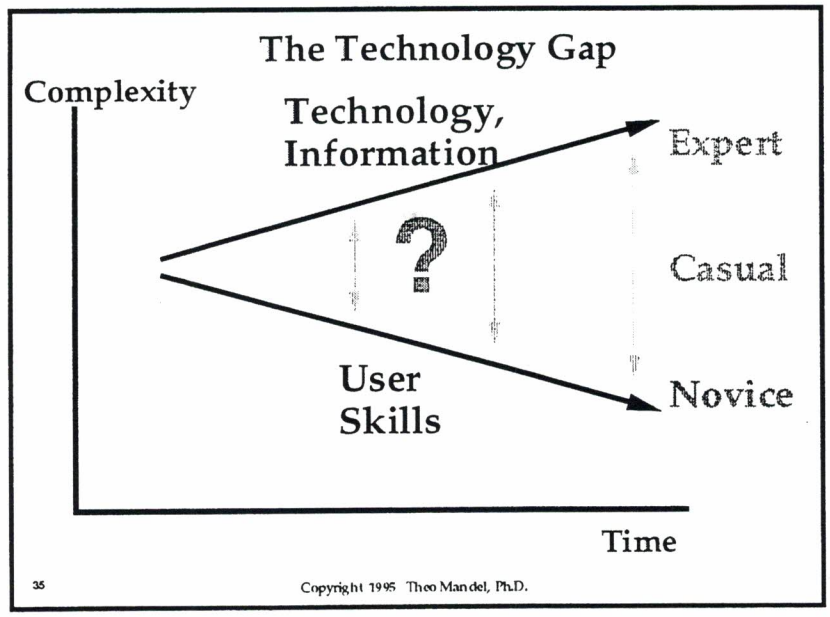
Future User Interfaces

- "I'll know the future is here when my left cufflink can communicate with my right cufflink via satellite."
  Nicholas Negroponte, MIT

- Computers are evolving:
  from
  Institutional systems
  to
  Personal systems
  to
  Intimate systems (Alan Kay, Apple Fellow)

34

Copyright 1995 Theo Mandel, Ph.D.



The Future

- GOAL: Amplify the human reach

We need to improve existing (and invent new):

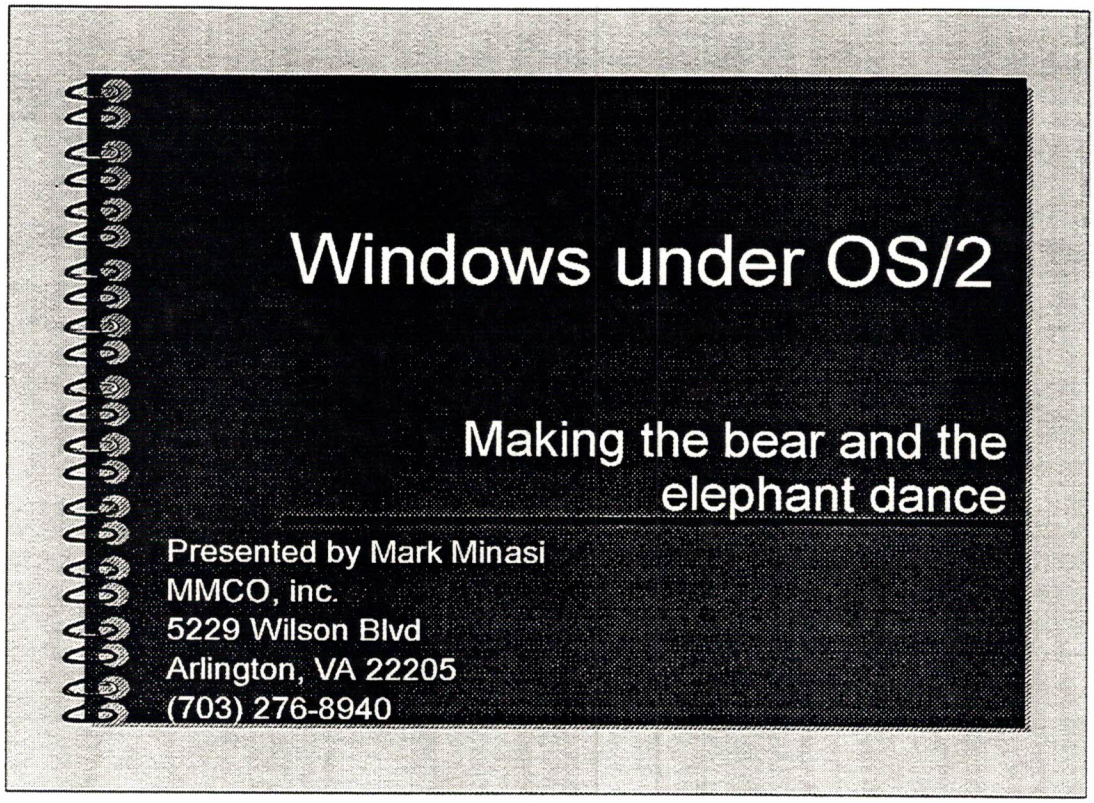
- Tools
- Interface and System Agents
- Interaction and Manipulation Techniques
- System and Interface Management Methods

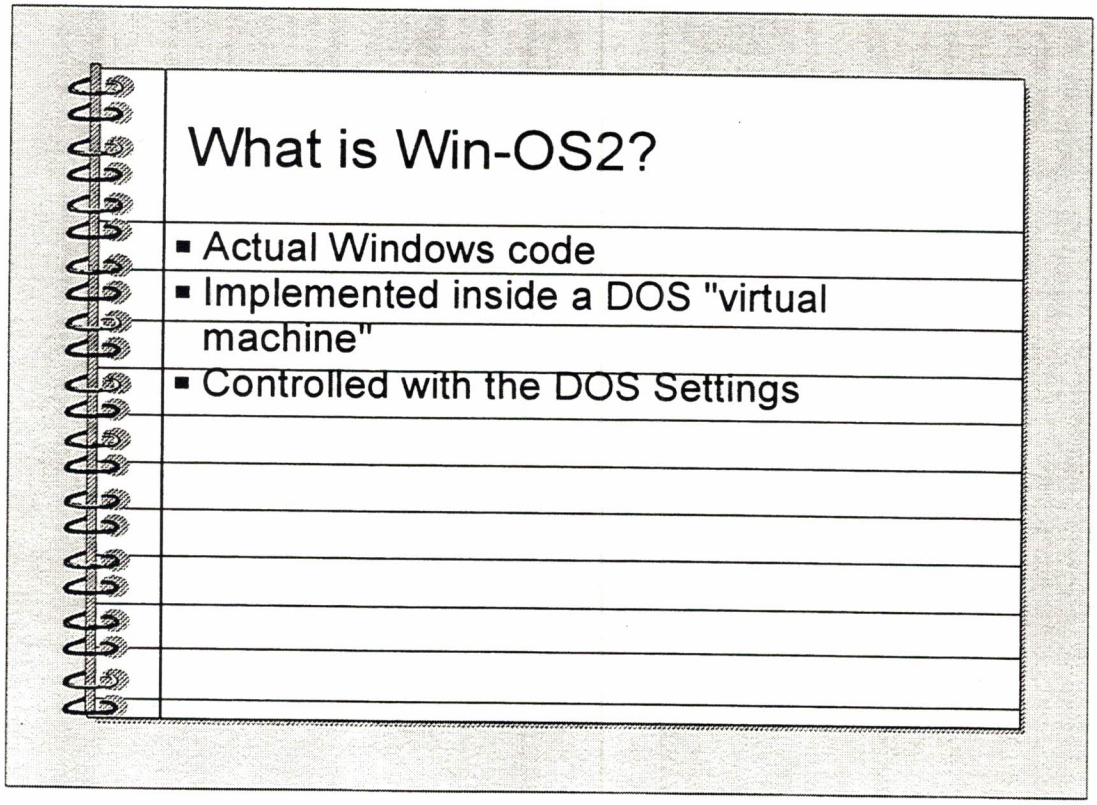
It's hard to design *simple* interfaces for *complex* technology and tasks

36 Copyright 1995 Theo Mandel, Ph.D.

SYS 0701
Configuring Windows Under OS/2

Mark Minasi
Mark Minasi & Company
Arlington, VA


Windows under OS/2
Making the bear and the elephant dance
Presented by Mark Minasi
MMCO, inc.
5229 Wilson Blvd
Arlington, VA 22205
(703) 276-8940


What is Win-OS2?
Actual Windows code
Implemented inside a DOS "virtual machine"
Controlled with the DOS Settings

Creating a Window object

- Drag a Program template from the Templates folder
- In the "Path and file name" field, include the name of the Windows program.
- Click the Session tab.
- Select "WIN-OS2 window," and check "Separate session."
- Adjust specific settings with the "WIN-OS2 settings..." button.
- You can access this from the settings notebook any time; some settings can be changed on the fly.

Important DOS Settings

- DOS_BACKGROUND_EXECUTION: set it off to save CPU time.
- IDLE_SENSITIVITY: set to 100 if the program locks up without warning.
- DPMI_MEMORY_LIMIT and XMS_MEMORY_LIMIT set the amount of RAM the program will see.
- COM_DIRECT_ACCESS and COM_SELECT are important for programs that use the serial port.
- MOUSE_EXCLUSIVE_ACCESS eliminates mice flying "in formation."

Making sure that Windows works

- As Windows sits atop DOS, the first test to do is to start a DOS session. If it works okay, then re-install Windows on Warp.
- Re-boot under DOS and make sure that your Windows installation works.
- Boot OS/2, open System Setup, and start Selective Install. Check WIN-OS2 Support, click More, and then indicate where Windows is (like C:\WINDOWS).
- Once Selective Install is done, you should run Windows fine.

Solving installation problems

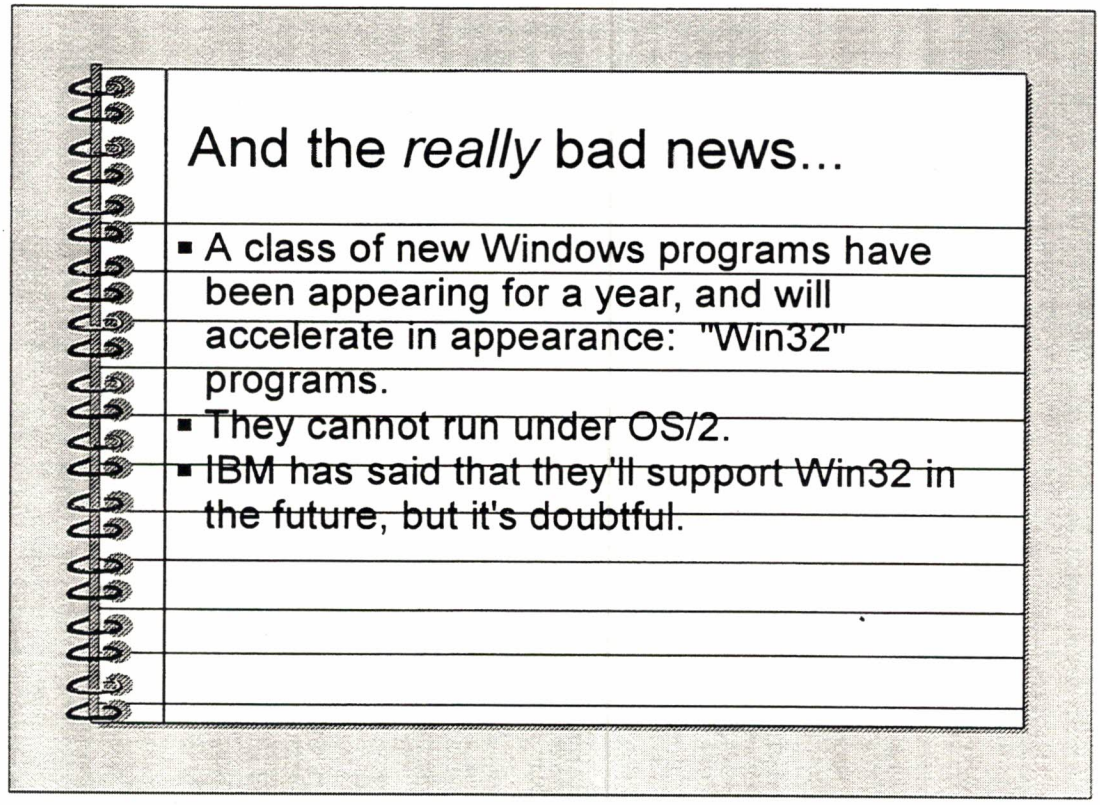
- OS/2 doesn't get along well with Windows for Workgroups; avoid it in a Warp environment if possible.
- You may have to re-install the system to get your drivers to work.
- Note that separate WIN-OS2 sessions will not share data except via the Clipboard or DDE, unlike normal Windows.
- Separate WIN-OS2 sessions will require a bit more memory, and may run a bit slower.

More install troubles

- You've got to install ATM fonts twice; once in OS/2, and once in Windows under OS/2.
- OS/2 doesn't use True Type fonts.
- Don't depend on DDE under OS/2 until you test it.
- The Migration Database often does a good basic job of starting up; look in the System Setup folder for the "Add programs" utility.

Be careful of file formats!

- Both Windows and OS/2 use file formats called WMF, BMP, WAV, and AVI, to name just a few.
- But the two types aren't compatible.
- Double-check compatibility if you're going to do anything cross-platform.


And the *really* bad news...
▪ A class of new Windows programs have been appearing for a year, and will accelerate in appearance: "Win32" programs.
▪ They cannot run under OS/2.
▪ IBM has said that they'll support Win32 in the future, but it's doubtful.

SYS 0710
Using OS/2 to Reach the Internet

David Gaertner
IBM

OS/2 WARP Topics

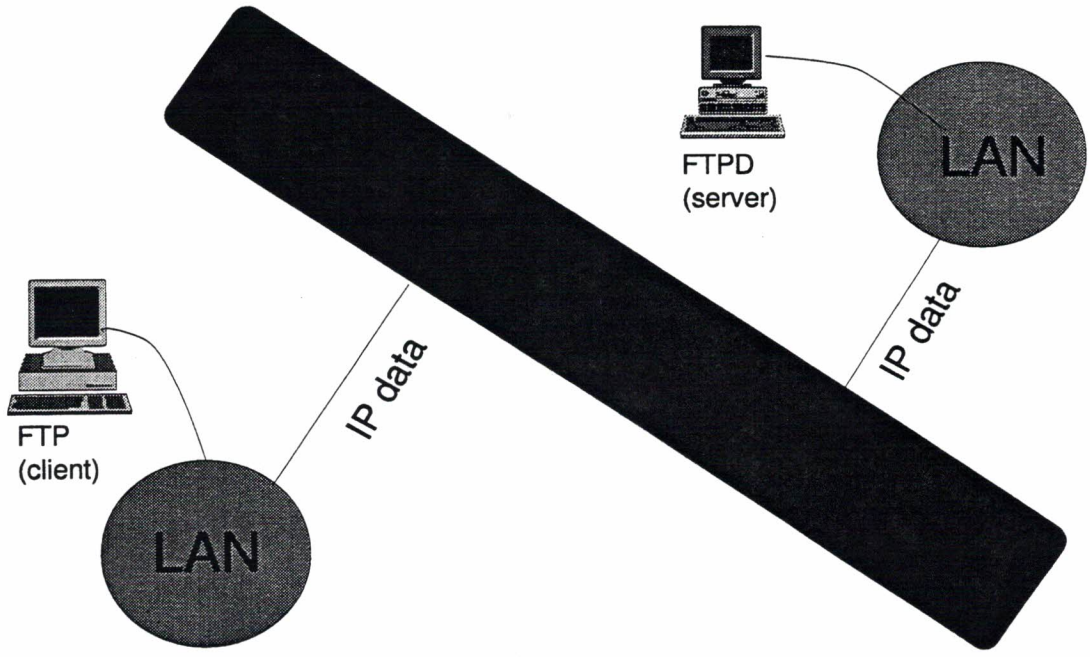
- Brief Introduction to the Internet
- Using the Internet Acccess Tools in OS/2 Warp and OS/2 Warp Connect
- Other Internet Tools for your OS/2 System
- IBM and OS/2 Warp Resources on the Internet
- Intra-Enterprise Applications of Internet tools

Please feel free to ask questions at any time, but please save the advanced technical questions for the end.

OS/2 WARP The Internet: Orbital View

- **Vast array of information resources**
  - Useful for businesses, hobbyists, professionals
    - *From air travel to computers to investments to zoology*
  - Internet is a good place to do many kinds of research
- **An important communication tool**
  - E-mail is by far the most used Internet application
  - Newsgroups often contain important information passed on by others
- **A Jumble of Old-fashioned and Modern Tools**
  - From inexpensive batch newreaders to resource intensive interactive multimedia user-interfaces
  - Still lots of UNIX! (ls, grep, dxwrxwr---, ...)

OS/2 WARP Internet: Linking LANs using IP



OS/2 WARP Origins of the Internet

- Original goal: share military research data
- TCP/IP was invented to make the Internet possible
- Originally mostly Unix-based
- IBM helped build the Internet
- Nurtured by researchers at military and universities
- Estimated number of users varies from 2 million to 30 million. Est. three million computers connected, world-wide
- Fact: number of users and Internet-connected information sources are currently growing rapidly

OS/2 WARP How Internet Can Be Useful to You

- **Communication with customers: e-mail**
- Retrieve files using FTP (file transfer protocol)
- Search databases using WAIS, Archie, & Veronica
- Share information with other users via newsgroups, gopher servers, and WWW servers
  - Newsgroups related to OS/2
    - comp.os.os2.advocacy, comp.os2.announce, comp.os.os2.apps, comp.os.os2.networking.misc, comp.os.os2.programmer.misc, et al
- Cautions:
  - Beware of viruses and scams
  - Use good *Netiquette*

OS/2 WARP Fun Things to Do

- **"Surf" - explore the Internet**
  - Use gopher or WWW clients (like Mosaic and the upcoming IBM WebExplorer)
  - Point and click to access Internet servers 'round the globe
- **Get on Mailing Lists**
  - Dilbert Newsletter: e-mail to `listproc@internex.net`
    - *Message:* `subscribe Dilbert_List Joe Blow`
- **Shop**
  - Order software, services, pizza, flowers
- **Get Federal, State, and Local government information**
  - lots of information made available on the Internet

OS/2 WARP Connecting to the Internet

- **Internet Access Types**
  - Via on-line services, shell accounts, SLIP and PPP, and even BBS
- **Internet Access Providers**
  - Canada: Hookup, NSTN, Onet, UUNorth, RISQ
  - IBM Global Network, PSI, AlterNet
  - America On-Line, CompuServe, Delphi, Genie, BIX
  - Many BBS's provide Internet e-mail and newgroup access
    - *Source: Krol (1994)*
- **Local providers can be discovered in in local newspapers and user group newsletters**

OS/2 WARP Some Internet Conventions

- **Internet addresses**
  - Computers on Internet have addresses E.g. 129.35.15.43
  - Most computers have associated names E.g. austin.ibm.com
  - Extension conventions:
    - `com`: business; `edu`: educational institutions; `gov`: government
- **Internet mail addresses**
  - Adress mail to a user on a certain computer
  - Often seen on business cards, increasingly in advertisements
  - Most common form: *user@node.ext*
    - `gaertner@austin.ibm.com`
- URL - Universal Resource Locator used on WWW
  - http://internet-name/{resource path name}
  - gopher, file, news

OS/2 Warp: Fast Path to the Internet

- **OS/2 Warp and Warp Connect are your ambassadors to the global community along the Information Highway**
  - BonusPak for OS/2 Warp functions include:
    - *IBM Gopher client software to make traveling the Information Highway 'mouse simple'*
    - *NewsReader to make the most productive use of your available reading time*
    - *E-Mail that can span the globe to maximize opportunities for you worldwide*
    - *FTPPM to exchange data and programs with a neighbor or someone in another country with the click of a mouse*
    - *Telnet to signon when you are remote from your server systems for increased flexibility and productivty*
  - OS/2 Warp Connect adds:
    - *Latest release of TCP/IP for OS/: 3.0*
    - *TCP/IP LAN access*
- **IBM provides you with productive access to the Internet from the very beginning**
  - Provides flexible user interfaces from GUI NewsReaders to command line Telnet access
  - Worldwide access services set up in minutes
  - IBM Global Network gives you the ability to join the global community
- **Leverage the 'International' in IBM**
  - IBM Information Highway 'on-ramps' available in major metropolitan areas worldwide
  - Maximize productivity with direct access to your office files from just about anywhere on Earth...

OS/2 WARP OS/2 Warp on the Internet

- **Easy Installation**
  - One Button Install
  - Pre-configured TCP/IP parameters
  - Dynamic creation of userid on-line at subscription initiation
- **IBM On-line Services**
  - BonusPak for OS/2 Warp Internet information highway access software
    - *NewsReader*
    - *UltiMail Lite*
    - *FTPPM & Telnet*
    - *Gopher*
    - *Dial access for OS/2 Warp*
  - Worldwide connection services directly to the Internet
    - *Servers provide access to the latest software and information*
    - *IBM WebExplorer beta can immediately be accessed and downloaded to make the most of the WorldWide Web.*
  - NewsReader/2 Subscriptions
    - *Family news subscriptions controled via master account password authorization*
    - *Over 8,000 news groups available*
  - Worldwide E-mail services
    - *Worldwide 'on-ramps'*
    - *Worldwide delivery in seconds*
    - *One account with up to 6 E-mail IDs for simple billing purposes*

OS/2 WARP IBM Worldwide Internet Services

- **IBM will put you in the fast lane of the Information Highway with high speed modem support**
  - Connection Services provided by IBM, from Hong Kong to Zurich to the OS/2 Internet Domain
- **OS/2 FTP site for the latest information**
  - OS/2 Games
  - OS/2 Utilities
- **OS/2 Technical Tips & Techniques**
  - OS/2 Development Information for Programmers
- **Worldwide E-mail**
  - Streamlined billing and exchange rate conversions
  - Up to 6 mail IDs per Account
- **Access thousands of databases worldwide**
- **Over 9,000 news groups**

OS/2 WARP Tools Used on the Internet

- **Older uses**
  - Mail
  - File transfer
  - Remote login
  - Conferencing
  - games
- **Newer applications**
  - Database search
  - "Cruising"
  - Multimedia crusing
  - chatting
  - games
- **Examples of tools**
  - mail, Ultimail
  - UUCP, ftp
  - rlogin, telnet
  - readnews, rn
  - chess by -mail
-
  - WAIS, archie, veronica
  - gopher
  - WebExplorer, Mosaic
  - Internet Relay Chat
  - MUD (multi-user dimensions)

OS/2 WARP Sources of Tools for Accessing the Internet

- **Native OS/2 TCP/IP**
  - IBM OS/2 Warp Version 3.0 - BonusPak
  - IBM OS/2 Warp Connect
    - *enables simultaneous LAN and Internet access*
  - IBM TCP/IP for OS/2
    - *for OS/2 2.11 users*
  - FTP Software's TCP/IP
  - Essex System's TCP/2
- **Windows tools**
  - Win32S and 16-bit versions work on OS/2 Warp
  - use WINSOCK to interface with TCP/IP
  - though some use include their own TCP/IP (e.g. Netcom)

OS/2 WARP Other Sources of OS/2 and Internet Tools

- **Walnut Creek Hobbes OS/2**
  - ftp://ftp-os2.cdrom.com/
- **MUENCHEN German OS/2 Archive**
  - ftp://ftp.informatik.tu-muenchen
- **NMSU OS/2 Area**
  - ftp://ftp-os2.nmsu.edu/pub/os2
- **Trosmo, Norway FTP Site**
  - file://ftp.cs.uit.no/pub/os2
- **Sweden's Archive**
  - file://ftp:luth.se/pub/os2

More Internet sites appear every week. See the handout for my latest list of Internet sources of tools, games, OS/2 Warp user information, and more.

OS/2 WARP Where to Find More Internet Information

- **Recent issues of OS/2 Magazine, PC Magazine**
- **Very many books on the Internet**
  - Krol, E. (1994). The Whole Internet User's Guide & Catalog, (2nd ed.). Sebastopol, CA: O'Reilly & Associates.
  - Gilser, P. (1994). The Internet Navigator (2nd ed.).
- On the Internet
  - http://www.ibm.com
  - http://www.ibm.net
  - http://www.austin.ibm.com/pspinfo
- IBM's Worldwide Electronic Resources
  - distributed by Vicci Conway (vicci@vnet.ibm.com) or CompuServe: 76711,1123

OS/2 WARP Intra-company Applications of Internet Tools

- **Intra-company and intra-enterprise Internets**
- **Internal WWW pages**
  - Reference information
  - Forms
- **E-mail**
- **Internal newsgroups**
  - product or service related discussion groups
- **Internal software repositories (via ftp, gopher, or WWW)**
  - internally developed software, data set, or tools
- **Connecting multiple sites**
- **Connections to the "real Internet" and security issues**
  - Firewalls

MGT 0809
The OS/2 Problem Solver

David Moskowitz
Productivity Solutions
Norristown, Penn.

The OS/2 Problem Solver

David Moskowitz

Productivity Solutions, Inc.

Agenda

- Installation
- Traps and application problems
- OOOPS stuff
- Printing problems
- How do I...
- Problem avoidance
- Worthwhile free tools
- Answer your questions

Installation

- Plan the installation
  - Failure to plan is planning to fail
- Partitions, Partitions, Partitions
  - OEM setup is **not** optimal
- Boot manager or dual boot
  - It is worth effort to use Boot Manager
- Verify you have the right & enough h/w
  - Sometimes parts get "borrowed"

Install problems

- Motherboard
  - Micronics motherboards revision E or later
- ROM BIOS
  - AMI roughly 1990 or later
  - Phoenix at least 1.02.05D
- White or blank screen
  - Turn off autosense
  - Force adapter into standard (VGA) mode
  - Force card into 8-bit mode
  - Use standard cheapo VGA for install

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

Using large disks

- Some BIOS already handle big IDE drives
- Bootable partitions totally contained in first 528M
- HPFS for the rest
- Ontrack
  - Setup Ontrack partition in DOS (use dynamic mode)
  - Select Advanced install in OS/2 and remove it with OS/2 FDISK and install

System doesn't boot

- Boot from maintenance partition to check boot volume, command line (Recovery choices) to check rest.
- Respond to error messages
- CHKDSK
  - Force CHKDSK at boot with +
- Cross linked EA's
- Rebuild INI files
- Make a case for Boot Manager
- Last resort reinstall

Errant application

- Ctrl-Esc to cancel errant applications
- Tool to kill app
  - WatchCat (Shareware)
  - PSPM (IBMEWS)
- SUPPRESSPOPUPS
  - POPUPLOG.OS2
- DUMPPROCESS
  - PDUMPxx
  - System appears dead during dump.

Trap xx

- TRAPDUMP=RO,c
  - SADUMP and proper size
  - REIPL=YES
- Record information of screen
  - Ctrl-Alt-NumLock-NumLock
  - Use CREATEDD
  - SAVEDSKF to make CREATEDD.DSK
    - Compresses to roughly 20K
  - Approximately 1 disk for each 2M RAM

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

Common problems

- Application aborts - Display registers, Print Screen or suppresspopups
- Application doesn't respond
  - Ctrl-Esc & end it; tell vendor to fix it
- Startup problems
  - LCtrl-LShift-F1 (2.1)
  - RESTARTOBJECTS
- Reset INI files
  - Alt-F1 (Recovery Choices)
  - Save your own

Common problems 2

- Win-OS/2 changes keyboard speed - KBD_RATE_LOCK
- Two mouse pointers - MOUSE_EXCLUSIVE_ACCESS
- Problems with COMMAND.COM
  - Verify path points to OS/2 version
  - Increase RMSIZE value
- System seems slow
  - Close applications you aren't using
  - Tune DOS session memory use (zap EMS memory if not needed)

Can't find COUNTRY.SYS

- Boot
  - Remove tape backup or similar devices
  - Have hard disk checked
- Normal operation
  - Could be real
  - CONFIG.SYS corrupted
  - CONFIG.SYS missing

Windows applications don't work

- Start Windows in a full screen DOS session
- useOS2shield == 0
- Path (DOS_AUTOEXEC)
- win.ini and system.ini settings
- DOS settings versus WIN-OS/2 settings
- Verify DRV files are in SYSTEM directory
- Not enough memory

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

Extended Attributes Problems

- CHKDSK for cross link pairs
  - DOS applications don't have EA's
  - ZIP files don't have EA's (yet)
- Copy one file of pair
  - Remove EA's from copy (EAUTIL)
    - for %i in (*.*) do eautil %i /s
  - Delete original
  - Restore from copy
- DOS file access can clobber EA's

OOPS

- Forgot desktop password
  - makeini os2.ini lock.rc (recovery choice)
- Save desktop arrangement
- Rearrange blues
  - Add a menu item with drag and drop
  - Open folders
    - SysSetObjectData folder, "OPEN=Default;"
  - Warp it (Undo arrange)
- REXX is your friend!

Printing problems

- Check printer - ON & port correct
- Proper driver and queue
- Forms mismatch
- IRQ related
  - Cable too long or missing wires
  - Interrupt conflicts
  - Adapter doesn't handle interrupts
- Sluggish or system not responsive - enable IRQ printing

Workplace Shell

- Enable desktop archive
  - If you move the archive, make sure the target has the same file system as the boot drive.
  - Don't leave it on -- it extends boot time
- Missing new scheme palette after update
  - From command-line issue the SCHEME command.

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

Backup Workplace Shell

- Manual method
  - BACKUP DESKTOP A: /s
  - copy config.sys autoexec.bat startup.cmd a:
  - copy \os2\os2*.ini a:
- Use LH2 or ZIP for large files
  - Preserves EAs
- Warp it!
- Other tools include Back Again/2, Deskman/2, WPSBkup, INIMaint suite, or XCOPY to a DCF/2 container

Single disk boot

- Only works with 3.5 inch disks
- Make it usable
  - Copy boot disk and remove files
    - OS2KRNL, OS2LDR*.* & OS2BOOT
  - Add ATTRIB, CHKDSK, FORMAT, UHPFS, XCOPY, etc.
- Use the same directory structure as OS/2
  - minus drive information (\OS2\DLL)
- Create utility disks and modify them

Maintenance Partition

- Basically the same system content as Single Boot Disk
  - Plus utilities & complete HPFS support.
- Separate 3M partition
- In an existing partition
  - Difficulty checking larger partition
  - No isolation
  - Works in a pinch
- IBMEWS BOOT2x automates creation

How do I...?

- Add support for Windows after installation
  - Install Windows under DOS
  - Use Selective Install in OS/2
  - Have your Windows disks handy
- Add DOS after installation
  - Dual boot to DOS and install
- Add Boot Manager to an OEM system
  - Backup, re-partition and format, restore
  - Partition Magic

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

How do I (2)...?

- Interrupt conflicts
  - RMVIEW
  - reserve.sys arguments
    - /irq: ; /IO:base,num ; /SHA ; /MEM:base,len
- Maintenance desktop
- Multiple config.sys files
  - ALTF1BOT, attrib and you.

How do I (3)...?

- Sound echos or repeats
  - Use RMVIEW to get hardware settings
  - Use Selective Install to make MM settings agree with hardware
  - Get updated driver from manufacturer
    - Sound Blaster users add SET BLASTER statement to VDM CONFIG.SYS
- Install new Windows drivers for full pack
  - PMDD_x disks (OS2IMAGE on CD-ROM)

How do I (4)...?

- Run games
  - See the README
  - Make sure you have proper screen resolution and palette
  - Add SET BLASTER to CONFIG.SYS
  - Check AUDIO_ADAPTER_SHARE
  - Many games are already in DATABASE.DAT
    - You can extend it

Steps to avoid or minimize problems

- Boot Manager
- Frequent backups
  - System
  - Desktop
  - INI and CONFIG.SYS
- Maintenance partition
- Remove unused fonts
- Plenty of space on the swap volume
  - Pre-allocte large swap file to avoid fragmentation on FAT and constant grow / shrink overhead

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

More problem avoidance

- Read the README's
- FTP transfer set BINARY mode
- Don't Stack your boot volume
- Add memory before you consider a processor upgrade
- Upgrade your video adapter before you upgrade the processor
- Upgrade motherboard to VLB and 72-pin SIMMs

Valuable free tools

- BOOT2X
  - Makes boot disk or maintenance partition
- MSHELL
  - GUI Workplace Shell replacement
  - No Object manipulation
- TSHELL
  - PROTSHELL replacement
- CMD.EXE replace either
- Look for useful REXX scripts
  - LISTINI, LISTEA, LISTINF, FIXOBJ...

Questions...

- The only dumb question is the one that isn't asked!
- I've never heard a dumb question.
- If not now, when?

Thank you!

For more information:

David Moskowitz
Productivity Solutions
164 Avondale Road
Norristown, PA 19403-2952
Voice: 610-631-5685 Fax: 610-631-0414
Internet: davidm2@ibm.net or
dmoskowitz@cis.CompuServe.com
CompuServe: 76701,100

Copyright 1995, Productivity Solutions, Inc. All rights reserved.

PRO 0801
Using Visual REXX to Produce Graphical Utilities

Bill Schindler
TGC
Cave Creek, Ariz.

Creating Utilities Using Visual REXX

Bill Schindler

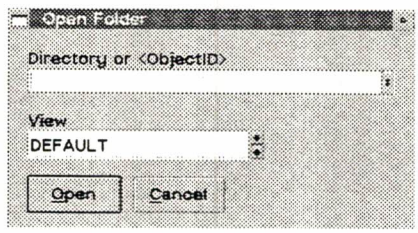
When designing a utility:

- Make the utility do one thing well.
- Think about how the utility will be used. Try to minimize mouse clicks and keystrokes.

An "Open Folder" utility using Hockware Vispro/REXX

Using a drive object to open a folder for a directory can be tedious if the directory is several levels deep. This is a simple utility that opens directory folders in a single step.

The first thing to do is to create the form. Here's what the finished program will look like:



The code to open a folder is fairly simple. The program needs to determine what to open and what view to use. Then it simply calls SysOpenObject() to open the folder in that view. The code attached to the **Open** button's *clicked* event:

```
Arg window self

/* Get item value ID_DIRECTORY Combo Box */
openWhat = VpGetItemValue(window,'ID_DIRECTORY')
/* Get item value ID_VIEW Spin Button */
view = VpGetItemValue(window,'ID_VIEW')

SELECT
  WHEN Left(openWhat, 1) = '<' THEN
    NOP; /* object ID does not require special processing */

  OTHERWISE /* must be a subdirectory */
    curDir = Directory()
    openWhat = Directory(openWhat)
    CALL Directory curDir
END

retcd = SysOpenObject(openWhat, view, 1)
IF \ retcd THEN
  retcd = SysOpenObject(openWhat, view, 0)
IF \ retcd THEN
  response=VpMessageBox(window,'Error','Cannot open "'openWhat'"')
ELSE
  DO
  /* Get index of value ID_DIRECTORY Combo Box */
  index=VpGetIndex(window,'ID_DIRECTORY','SEARCH',openWhat)
  IF index = 0 THEN
```

```
    DO
    /* Add an item at beginning ID_DIRECTORY Combo Box */
    index=VpAddItem(window, 'ID_DIRECTORY', 'BEGIN', openWhat)
    END
  END
```

When the form is opened, we initialize the different values. Since we're using an INI file to remember the last twelve folders, we need to find the INI file and load the directory names. The *When opened* event for the form is:

```
Arg window
/* Load RexxUtil */
CALL RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
CALL SysLoadFuncs
/* Add items from list ID_VIEW Spin Button */
value.0 = 2
value.1 = 'DEFAULT'
value.2 = 'SETTINGS'
index=VpAddItem(window, 'ID_VIEW', 'LIST', 'VALUE.')

PARSE SOURCE . . myName
dirName = FileSpec('D', myName) || FileSpec('P', myName)
iniFile = dirName || 'openfldr.ini'

count = SysIni(iniFile, 'Names', 'Count')
IF count >< 'ERROR:' THEN
  DO
  DO i = 1 TO count
    value.i = SysIni(iniFile, 'Names', i)
  END
  value.0 = count
  /* Add items from list ID_DIRECTORY Combo Box */
  index=VpAddItem(window, 'ID_DIRECTORY', 'LIST', 'VALUE.')
  END
```

When the folder is closed, the INI file needs to be updated. The *When closed* event for the form is:

```
Arg window
/* Get item count ID_DIRECTORY Combo Box */
count=VpGetItemCount(window, 'ID_DIRECTORY')
count=Min(count, 12)
CALL SysIni iniFile, 'Names', 'Count', count

DO i = 1 TO count
  /* Get item value at index ID_DIRECTORY Combo Box */
  value=VpGetItemValue(window,'ID_DIRECTORY',i)
  CALL SysIni iniFile, 'Names', i, value
END
```

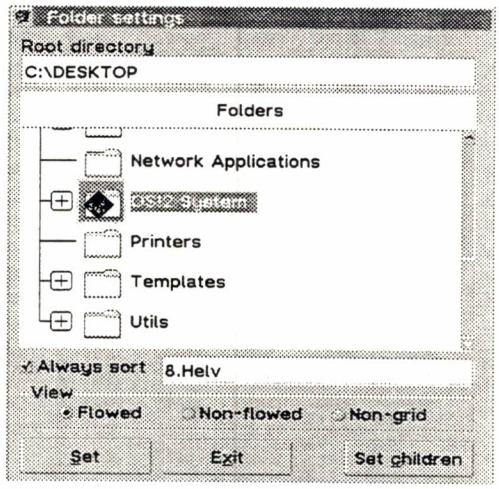
A "Folder Settings" utility using Watcom VX-REXX

Another tedious job is changing settings for several folders on the desktop. This utility is an attempt to simplify that job by allowing you to change settings for several folders at once.

Folders on the desktop are actually subdirectories in the desktop directory. One of the first problems to solve is how to find the desktop directory. The method used in this code works about 99% of the time.

```
Window1_Create:
/* Load RexxUtil */
```

```
CALL RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
CALL SysLoadFuncs

/*
 * Get the current desktop location
 */
CALL SysIni 'SYSTEM', 'FolderWorkareaRunningObjects', 'ALL:',
      'workarea.'
desktopDir = workarea.1

ok = VRSet( "EF_DesktopDir", "Value", desktopDir )
ok = VRSet( "RB_2", "Set", 1 )
ok = VRSet( "CN_Folders", "ExtendSelect", 1 )

/*
 * Get all the folder names
 */
CALL SysFileTree desktopDir'\*', 'desktop.', 'DOS'

rootHandle = VRMethod( "CN_Folders", "AddRecord", , "First",
      FileSpec('N', desktopDir), desktopDir, ,  )
dirDrive = FileSpec('D', desktopDir)
CALL LoadFolders 1, FileSpec('P', desktopDir'\x'), rootHandle

RETURN
```



The work of actually changing the folder's settings is performed by the code attached to the **Set** button. The user's desired settings are read and a settings string is built. The SysSetObjectData function is called to change the settings for the folder.

```
PB_Set_Click:
doSort = VRGet( "CB_1", "Set" )
isFlowed = VRGet( "RB_1", "Set" )
isNonFlowed = VRGet( "RB_2", "Set" )
isNonGrid = VRGet( "RB_3", "Set" )
fontName = VRGet( "EF_FontName", "Value" )

IF doSort THEN
  settings = 'ALWAYSSORT=YES;'
```

```
ELSE
  settings = 'ALWAYSSORT=NO;'

SELECT
  WHEN isFlowed THEN
    settings = settings || 'ICONVIEW=FLOWED;'
  WHEN isNonFlowed THEN
    settings = settings || 'ICONVIEW=NONFLOWED;'
  WHEN isNonGrid THEN
    settings = settings || 'ICONVIEW=NONGRID;'
  OTHERWISE
    NOP
END

IF fontName >< '' THEN
  settings = settings || 'ICONFONT=' || fontName || ';'

ok = VRMethod( "CN_Folders", "GetRecordList", "Selected",,
               "selected." )
DO i = 1 TO selected.0
  DO j = 1 TO desktop.0
    IF selected.i = desktop.j.!handle THEN
      DO
      fullDir = desktop.j
      LEAVE j
      END
  END j
  IF \ SysSetObjectData(fullDir, settings) THEN
    DO
    id = VRMessage( VRWindow(),,
                    "Error changing settings for" fullDir, "Error",,
                    "None",,,  )
    END
END

return
```

PRO 0802

Lotus Notes as a Development Tool (Using REXX)

Brian Proffit

Visionary Reseach

Boynton Beach, Flor.

According to figures from Lotus Development, nearly one million people are experiencing the joys of groupware via Lotus Notes. There can be no doubt that Lotus has created a new category of application, and one for which the market has found good use.

Notes brings with it some old baggage, though. The interface you use to develop Notes applications is based loosely on the same macro language used by Lotus 1-2-3 over a decade ago. Admittedly it has been extended quite a bit since then, but it is still clearly designed to make it easy for Lotus to interpret your instructions rather than being easy for you to write them. Fortunately, Lotus provided a back door through which you can make a connection with external programs. That can make it easier for anyone to enhance their Notes applications.

WHY DO I CARE?

Despite its richness, the Notes interface is designed to provide "behind the scenes" processing of your information, rather than letting the aggressive user expand their application into more graphical or interactive functions. For example, one of the most common uses of Notes is to maintain a database of people -- whether customers, contacts, or vendors.

The need for databases like this has been clear for years, and Notes has helped make them more visually usable and sharable -- even across operating platforms -- than ever before. On the other hand, how many people would buy a Personal Information Manager (PIM) these days that didn't provide an autodialer? I won't argue the social aspects of people becoming so accustomed to changing television channels without leaving their chair that the act of pushing the phone buttons is an annoyance.

The phone numbers are in the computer and the modem is attached, so why not get the extra productivity? Notes doesn't provide an interface to allow all of those people-oriented databases to connect to the telephone. But we have our ways. . .

@ YOUR COMMAND

The secret is the @command function. This function can instantly transfer control to any program you want and, in a limited way, send that program some data. Let's take a look at how you can use this function to add auto-dial capability to your existing Notes applications.

Open a contact-related Notes application (for which you have Designer privileges) and select the menu option Design / Forms to bring up a list of the forms defined for that database. Select a form which shows an individual's name and phone number, and click on Edit. Move the cursor down to the end of the field containing the phone number. Now tab to the right, and select the menu option Edit / Insert / Button. Notes will respond with the button assignment dialog, and your screen will look similar to Figure 1.

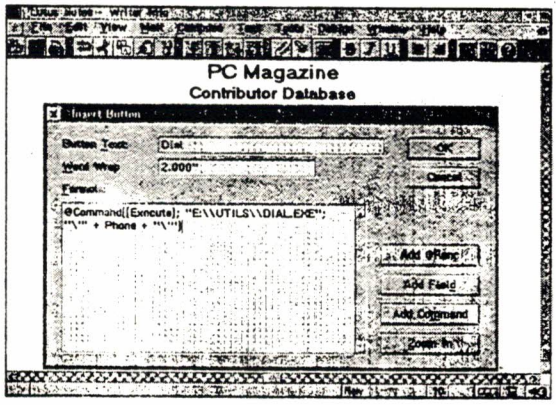


Figure 1: You assign characteristics to a button on a nOtes form through this dialog box

As you can see, we've put the word "Dial" onto the button to indicate its purpose. The formula that is executed when the button is pressed needs a little more explanation.

ASSIGNING A PROGRAM TO THE BUTTON

The main purpose of the @command function is to execute options from the Notes menus as if the user had performed the selection herself. The argument inside the square brackets indicates the menu item to perform. In this case,

though, we've activated the Notes back door by specifying that we want to Execute the next argument as if it was typed at a command line.

Note that the backslash ('\') is used by Notes to indicate things like cascaded menus, so in order to describe the path to the program you must use double backslashes to indicate subdirectories. We'll build the dialing program shortly.

The third argument defines the parameters to be sent to the program as if they were typed after it on the command line. Because of the limitations of Lotus' command interpreter, though, Notes only officially supports a string constant here, such as the name of a file the designated program should load. In our case, though, we want to send the program data from our application -- the phone number we want to dial.

The convoluted string in Figure 1 allows us to circumvent Notes' limitation. The first part is a quote, a backslash, an apostrophe, and another quote. I said earl ier that Notes treats the backslash as a special character. In this case, it marks the start of a string, and allows more characters to be concatenated to the string. If you were passing a constant phone number, you would simply put it between quotes without using the backslash, i. e. "800-555-1212". The backslash tells Notes that a field's value ra ther than a constant will be inserted into the string.

The apostrophe acts as the opening delimiter of the parameter to be passed. Whatever appears between the apostrophes (after the field values have been inserted) will be treated as if it were a constant string. Next, we concatenate "Phone", the name of the field whose contents we want to send to the program. Finally, we terminate this with another quote, a backslash, an apostrophe to end the phone number string, and a final quote to end the argument. Hey, if it was easy you wouldn't need to read PC Magazine to find out how to do it!

You can extend this technique to pass more than one value to your program. Suppose we also wanted to pass the writer's name to appear in the message box. We could do so by making the third argument:

```
"\"' + Phone + "," + WriterName +
"\'"
```

The program would then receive the two fields separated by a comma so they could be pulled into different variables.

That's all there is to it on the Notes side. Click on OK to accept the button, then press Escape to get out of edit mode. If you have a form of this type open, you will have to close it and reload before your changes will be seen. You should then have a screen similar to that in Figure 2.

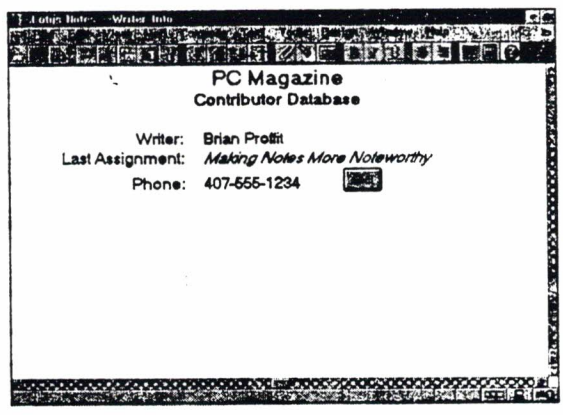


Figure 2: The Dial button adds an easy-dial function.

Note: There was a bug in Notes 3. 1. 5 for OS/2 that caused it to not pass arguments to the program being executed, so this technique will not work with that version. It does work, however, with earlier and later versions of the OS/2 software, and with all versions of Notes for Windows.

PROCESSING THE PHONE NUMBER FROM NOTES

We need to create a program to actually dial the phone. The program must have some level of interactivity, too. It must tell the user when to pick up the handset, it must prompt them to

indicate that they have done so, and then it must hang up the modem line. When you call your parents, do they both get on different extensions to talk to you? The voice quality from both of them suffers. We need to hang up the modem so that your handset won't suffer the same problem.

This is all a bit more awkward to do from a batch file, and it requires the operating system to start a command session. That is ugly and confusing to many users, particularly under Windows if you haven't changed the default to a windowed session and your Windows desktop disappears! It's much smoother to write a quick program that allows all this to appear to the user as a natural part of Notes. {{ I don't think the "essentially impossible" judgment is supported here. To execute a DOS batch file, you'd need to run the program C:\DOS\COMMAND. COM and pass the string "/C BATNAME. BAT (800)555-5555" to it. Then we could build a batch file and execute it w/o worrying about command-line parameters. Besides, this gives us a chance to dress it up and make it look professional.

We'll write this program using Visual Basic for the Windows people, and using VX-REXX for the OS/2 world. The technique is remarkably similar regardless of the tool being used -- in fact the first part of the task is essentially identical as you can see in Figures 3 and 4.

We've created a small dialog box as the only form in the application. The large blank area on the left side of the form is used to fancy things up a bit -- it is a bitmap space which I've assigned to an icon of a telephone. VX-REXX doesn't show the bit map until the program is run. I've broken the top line of text into two parts; one with the words "Connecting with", and a second text field where I'll plug in the telephone number when the program is run. Remember to define the first text field as right-justified so the words run together like a normal sentence when the phone number is put into the second field. The second line instructs the user to pick up the handset and then close the dialog. That's important, because that's the key for the program to hang up the modem. There is a standard pushbutton at the bottom of the dialog. This button has been defined as the default action, so either pressing Enter or clicking on the button will close the dialog.

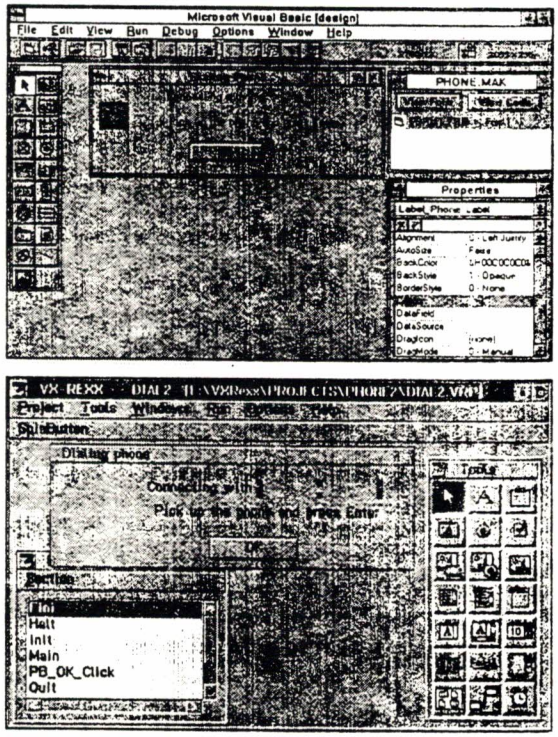


Figure 3: The first step in creating a dialog is essentially the same in Visual Basic (top) as it is in VX-REXX (bottom)

Also, we want this to appear as a modal dialog to Notes itself. It is typical to remove the minimize and maximize icons from that type of window, as I've done here. I've also changed the window's border from the normal thick sizable frame to a thin fixed-size frame. One other change I've made in the Properties window is to set the ControlBox to False in Visual Basic and uncheck the System Menu box in VX-REXX. This forces the user to

```
Sub Form_Load ()

'** Pull phone number from command line and strip of apostrophes
Phone$ = Mid$(Command$, 2, Len(Command$) - 2)

'** Prefix with "1" if long distance
If (Len(Phone$) > 8) And (Left$(Phone$, 1) <> "1") And (Left$(Phone$, 3) <>

'** Now initialize the modem and dial the number
Open "COM2" For Output As #1
Print #1, "ATDT" & Phone$

'** Finally, put the number in the dialog and display it
Label_Phone.Caption = Phone$

End Sub
```

Figure 4: The Visual Basic code is attached to the form

click on the pushbutton to trigger hanging up the modem. You can see these changes with VX-REXX, but Visual Basic doesn't show them until the program is run.

ASSIGNING CODE TO THE FORM

The program steps that need to be attached to the form are naturally a little dif ferent between Basic and REXX. Let's look at the Windows side first. Double-click on a blank area of the form to define instructions to be executed before the dialog is displayed, as shown in Figure 5. [[insert ea22-5. pcx - "Visual Basic code to dial the phone"]] Starting on the line under the Sub Form_Load () statement, enter the following instructions: (Note: the numbers at the start of each line are for illustration purposes only, and should not be entered. The lines that begin with an apostrophe are comments and may be entered but are not necessary.)

Visual Basic automatically puts any arguments passed to it from the command line into the predefined variable Command$. Line 2 extracts the phone number from Command$, removing the apostrophes we had to put around it in Notes. Line 3 checks to see if this is a long distance number. If it is, it checks to see if there is already a "1" at the front of the number, or an "011" for international calls. If needed, line 3 will prefix a "1" to

```
'** Define variables
 1 Dim Phone$
'** Pull phone number from command line and strip of apostrophes
 2 Phone$ = Mid$(Command$, 2, Len(Command$) - 2)
'** Prefix with "1" if long distance
 3 If (Len(Phone$) > 8) And (Left$(Phone$,1) <> "1") And (Left$(Phone$,3) <>
"011") Then Phone$ = "1-" & Phone$
'** Now initialize the modem and dial the number
 4 Open "COM2" For Output As #1
 5 Print #1, "ATDT" & Phone$
'** Finally, put the number in the dialog and display it
 6 Label_Phone.Caption = Phone$
```

Figure 5: This is the Visual Basic code for dialling the telephone

the front of the number. Line 4 opens the communications port so that we can send commands to the modem. If your modem is on a port other than COM2, change the line accordingly. Line 5 does the actual dialing, sending the modem command to take the line off hook, wait for a dial tone, and then dial the number. Note that if you don't have touch-tone phone capability, you should change the command to "ATDP" f or pulse dialing. Finally, line 6 copies the phone number into the blank field on the form so it will appear in the dialog. If you named the blank field something other than Label_Phone, use the appropriate object name.

We're halfway home; we just need to process the Enter/pushbutton press and end the program. Move to the top of the code window, and change from the Form object to whatever you named the pushbutton and the Click procedure. Insert the following instructions (again, without the line numbers):

```
1 Print #1, "ATH"
2 Close #1
3 End
```

The first line sends the modem a command to hang up the phone. Don't worry, your handset will stay connected. The second line closes the communications port, and the third line tells Visual Basic to end the program. Now save your project, and select the option from the File menu to create an EXE file. Be sure to use the same name that you used in the Notes @Command statement.

That's all there is to it! Now go back to Notes, and open a form containing the new Dial button. Click on the button, and listen for the dial tone as your system responds as shown in Figure 6.

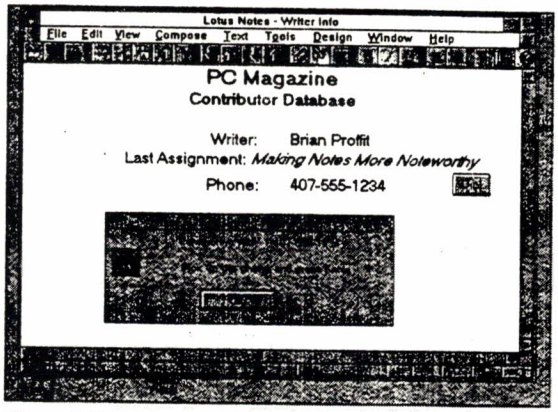


Figure 6: **After you've attached code to the form, you can dial automatically from your Notes database.**

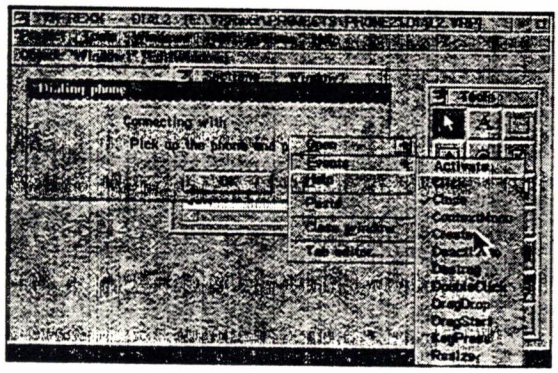


Figure 7: **In VX-REXX you assign code to the form using this pop-up menu.**

VX-REXX IS JUST AS EASY

OS/2 users will find the VX-REXX code just as straightforward as the Visual Basic effort was. Click with the right mouse button on an empty area of the form to bring up the form's pop-up menu.

Click on the arrow beside Events for a list of the events that can occur on the form, as shown in Figure 7.

Click on Create to open a window to enter the instructions to execute before the dialog is displayed. Move to the line underneath the Window1_Create label and insert the following instructions (without the line numbers):

VX-REXX copies command line arguments into a compound variable named InitArgs. Line 1 extracts the phone number and strips the apostrophes. Lines 2 through 4 are almost exactly the same as their Visual Basic counterparts, and handle the same long distance checking and phone dialing functions. Line 5 puts the phone number into the empty text area of the dialog. If your modem is attached to serial port other than COM2 or you named the text field something other than DT_Phone, you will need to adjust the program accordingly.

Now finish the program by adding the code to execute when the pushbutton or the Enter key is pressed. Right click on the pushbutton to bring up its pop-up menu, and select the Click event. Move the cursor to the line underneath the OKButton_C lick label and insert the following instructions (without line numbers):

```
1 call LineOut 'COM2', 'ATH'
2 call Stream 'COM2', 'C', 'Close'
3 exit
```

These lines correspond exactly to their Visual Basic counterparts to hang up the modem, close the serial port, and end the program. Now save the project and create an executable file with the same name you used in the Notes @command function. The results will be exactly the same as those in Figure 6.

NOW LET YOUR IMAGINATION RUN

This example adds a very valuable function to your Notes database, but it's only the beginning. Now that you know how to assign a program to a pushbutton and pass it data from your Notes form, you can probably come up with dozens of ways to make Notes even more valuable for you.

```
1 Phone = Strip(InitArgs.1, , "'")
2 If Length(Phone) > 8 & Left(Phone,1) <> 1 & Left(Phone,3) <> '011' t
Phone = '1' || Phone
3 call Stream 'COM2', 'C', 'Open'
4 call LineOut 'COM2', 'ATDT' || Phone
5 call VRSet 'DT_Phone', 'Caption', Phone
```

Figure 8: This is the VX-REXX code for dialing the telephone

What do you do when you need to access the information in a Lotus Notes database without actually starting Notes? With the right tools there is a way to access Notes databases in your own programs—just as if they were regular data files.

In our issue of January 24, 1995, we examined ways to enhance your Notes applications by attaching an external program to a push button on your Notes forms. We added a push button to a Notes document so you could automatically dial the phone number of a person whose name is contained in the document. It would be even more convenient, though, to develop a general solution that didn't require starting Notes, opening the appropriate database, and then opening the document for that particular person—all just to dial the phone.

In this issue, using Brainstorm Technologies' VB/Link for Lotus Notes and Microsoft's Visual Basic for Windows (or IBM's InterfloX and Watcom's VX-REXX for OS/2 users), we'll build upon that earlier program to create a new utility that reads the Notes database and displays a list of contacts and their phone numbers. You can then select the desired person and dial his number without having to run Notes itself. The code for the programs in the previous and present articles can be downloaded from PC MagNet's Programming forum, archived as DIALER.ZIP.

A NEW VBX CONTROL VB/Link is a set of Visual Basic (VB) functions that allow access to and manipulation of Notes databases through its new NotesData control for VB. The control acts primarily as a focal point for calling methods to read and write Notes data. While most of its functions are strictly for getting Notes data to and from Visual Basic objects, it also contains functions for piping Notes information directly into VB grids or list boxes.

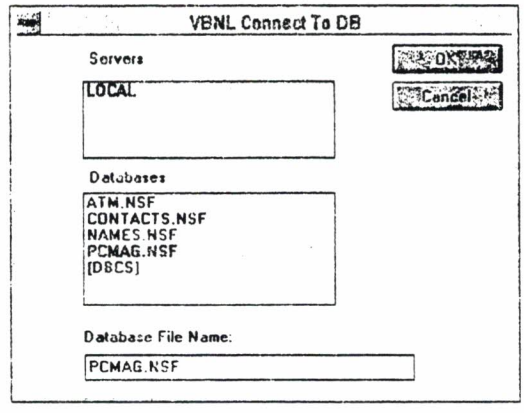


Figure 1: **This is the VB/Link dialog that provides a direct connection to Notes databases.**

So, begin a new project and create a grid nearly as wide as the form, leaving a little space at the top and an inch or so at the bottom of the form. Next, draw a rectangular area to hold the new NotesData control in the lower left corner of the form. Size doesn't matter here, and you should set the Visible property to False.

The control must now be attached to the Notes database. In the Properties dialog, double-click on Database. As you can see from Figure 1, you'll be presented with a dialog very similar to the one Notes itself uses for selecting a database. Double-click on the desired server to get a list of the available databases, and then double-click on the desired database to make the connection. Now specify which form within that database you want to access. Double-click on the Table property, and the drop-down list box at the top of the properties window will allow you to select the specific Notes form you need.

To complete the utility window, add a command push button centered at the bottom of the form and change its caption to "Dial". You'll probably also want to change the caption for the form to something like "Contact Telephone List.". Your form should look similar to the one in Figure 2.

PUTTING MEAT ON THE SKELETON The next step is to attach code to the objects you've just created. Double-click on an empty area of the form to bring up the editor, and then enter—without the line numbers—the following Form_Load procedure:

```
1 Dim ret%
2 NotesData1.Action = ACTION_LOGON
3 NotesData1.Action = ACTION_RUNQUERY
4 ret% = VBNL_PipeNotesToGrid(Notes
  Data1, Grid1, 2)
5 NotesData1.Action = ACTION_CLOSEQUERY
```

The first line declares the variable that will hold return codes. The second and third lines perform methods that VB/Link provides with the NotesData object: Line 2 logs onto the contact database, and line 3 runs a query against the view specified in the Table property. VB/Link mirrors Notes data in *result sets,* which are essentially tables in which each row is a separate document and the columns correspond to the fields in those documents. This correlates well with the VB grid control. The fourth line sends the results of the query directly into the grid you drew, and the fifth closes the connection to the Notes database.

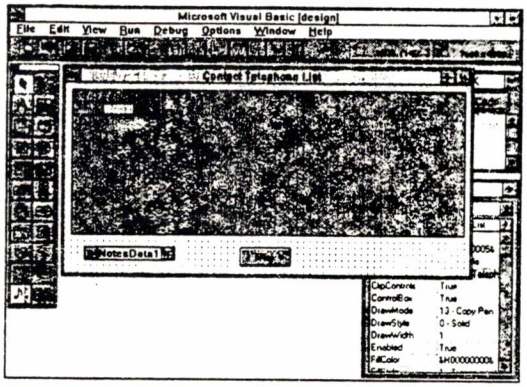


***Figure 2:* The NotesData control is on the left side of the phone number selection form.**

Once a user has selected a person, he will click on the Dial command button. For completeness, though, you should put the following line of code in the (default) Grid1.DblClick procedure:

```
call Dial.Click [Repl.
```

to call the button's click routine. This gives a user the choice of simply double-clicking on the contact rather than clicking on the person and then on the Dial button.

The next step is to create that button code. Double-click on the command button to bring up the editor, and enter—without the line numbers—the following Command1_Click procedure:

```
1 Grid1.Row = Grid1.SelStartRow
2 Grid1.Col = 2
3 Phone$ = Grid1.Text
4 Form2.Show 1
```

Since the grid control is a table, the user could select any rectangular area. The first line restricts calling to one number at a time by specifying the first row selected. Line 2 picks the third column (numbering starts with 0) in that row. Note that the user could have clicked on the person's name, but this procedure moves the focus to the contact's phone number. Line 3 extracts the phone number into the variable Phone$. (Yes, I know we haven't declared the variable, but I'll get to that in a moment.) Line 4 causes Visual Basic to display the second form as a modal window.

You don't remember creating a second form? Okay, I cheated a little. Use the form created in the January 25, 1995, Notes extension story. Save it as Form2 in the directory where you're creating this new project. Then add that form to the new project. The window displays the phone number being dialed and prompts the user to press a button when the handset has been picked up so that the modem can be disconnected. To do that, though, you've got to get the phone number to the new form. Variables declared within a form's code are visible only within that form, so you must define Phone$ as a global variable, making it visible to all forms.

From VB's File menu, select New Module. The editor will open a new window for the declarations portion of Module1.BAS. Enter the line

```
Global Phone$
```

Now the phone number can be used by both forms. You'll need to double-click on an empty area of the second form, though, and modify its Load procedure.

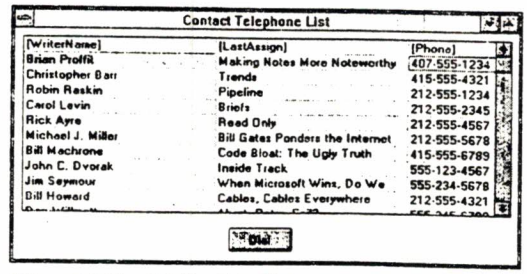


***Figure 3:* With a list like this, Windows users can easily dial any contact from a Notes database.**

You must remove the Dim Phone$ line from that procedure. You must also remove the line that extracts the phone number from the command-line arguments, since you're getting it directly this time. The remaining code (to prefix a 1 for long distance and actually to dial the number) works fine as it is.

There is one other change to the second form's code. The previous button-click routine contained an End statement to stop the program after hanging up the modem. Change that line to

```
Unload Form2
```

This will close the form and return the user to the contact list. Now save the project and create an .EXE file from it. You can add the new program to your desktop as a standalone contact dialer. Running the program should bring up a display similar to that shown in Figure 3. As you can see, VB/Link inserts the names by which Notes knows the fields into the first row to serve as column headings.

FOR OS/2 USERS TOO The OS/2 world can access Notes databases outside Notes as well, albeit through a slightly more convoluted procedure. IBM made a different design decision when developing InterfloX. Rather than depending upon a graphical development tool (as VB/Link depends on Visual Basic), InterfloX provides Notes access to any REXX program—including those that are totally character based. That is a real advantage for some purposes, but it has a downside as well. Whereas VB/Link provided simple development through a VBX control, InterfloX requires more complex programming to display Notes information. You could use any of a number of popular REXX tools to do the job, but for this particular example I'll be demonstrating with VX-REXX.

Open a new project and narrow the form to roughly a square. Move the pointer to the title bar, and while holding down the Alt key, click the left mouse button. The title bar will become a text-entry field, allowing you to change the window's title to something more meaningful like "Telephone Dialer".

Now create a list box that is nearly the full width of the form and extends down from the top to about an inch from the bottom. Then put in a push button that is centered underneath the list box. Use the Alt-<Left Mouse Button> trick so you can change the text on the button to the word "Dial." Your form should now look similar to the one depicted in Figure 4.

BRINGING IT TO LIFE Double-click on Init from the Section List to open the editor with the code that is executed when

the program is initialized. Move the cursor to the line below

```
call VRSet window, "Visible", 1
```

and enter—again, without the line numbers—the instructions listed in Figure 5.

The first line in Figure 5 simply initializes the Notes/InterfloX environment. The second line opens the specific Notes database you'll be working with and returns a database ID—a handle by which to access that database.

Actually accessing the contents of that database is an indirect process. Rather than simply reading through the documents serially, you must specifically request them using a document ID. Line 3 conducts a search of the database, selecting only those documents that represent the form in which you're interested. In addition to your data documents, the access control list used to determine who can read or change the information is also stored in document form in the database itself. So are the instructions on the views that are available. Line 3 extracts only our specific form type and creates a new compound variable (*writer* in this instance) that contains the document IDs that reference those documents. InterfloX follows the REXX standard of using element 0 of a compound variable to contain the total number of elements in the array.

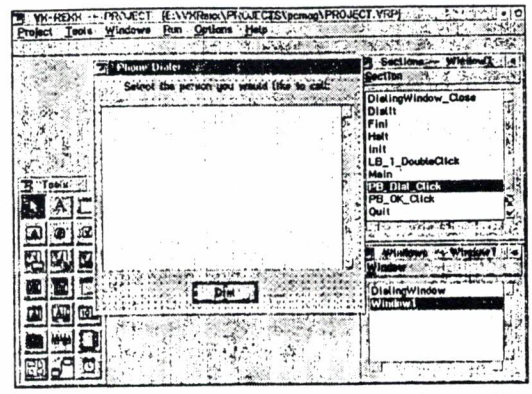


Figure 4: **This is the phone-dialer form used when working with VX-REXX.**

Line 4 begins a loop that steps through the document ID array and uses those handles to read the specific documents that they represent. Line 5 performs that read and puts all of the contents of the current document into a new compound variable called *docinfo*. The structure of this variable is described in Figure 6.

Having loaded the document, we now need to look through it for the name and phone information. Line 6 begins a loop through the fields found in the document whose layout is described in Figure 6. I use this technique rather than specifying field numbers, because forms change all the time. By looking for specific field names in lines 7 and 8, the program will continue to work even if new fields are added, removed, or inserted later. The only assumption the program makes is that a contact's name appears before his phone number on the form.

Line 7 finds the contact's name and stores it for later usage. Line 8 detects that the current field is the phone number and exits the loop. Line 10 puts an entry in the list box consisting of the name we saved, a dash (a separator), and the value of the current field (the phone number). Finally, line 12 closes the Notes database and line 13 terminates the Notes/InterfloX connection.

MAKING THE CALL We now need to process the phone number the user selects when the Dial button is pressed. Click on Dial with the right mouse button, and select Click from the Event menu. This brings up the editor to allow us to enter instructions for the PB_1_Click event. Enter the following instructions, again omitting the line numbers shown:

```
1 call VRMethod 'LB_1',
  'GetSelectedStringList', 'person.'
2 parse var person.1 . '-' phone
3 call DialIt phone
```

Line 1 determines which entry the user picked and puts it into the variable *person*. Line 2 then parses that entry, using the dash we inserted as the delimiter that indicates the start of the phone number. Line 3 calls the DialIt procedure, passing it the phone number that we want to call.

At this point, we must again cheat as we did with the Visual Basic version and use the window and program that we created in the earlier article to display the phone number and make the call. Copy that window as Window2 and, since it is no longer the primary window, rename the Window1_Create routine DialIt. Only a couple of small changes are needed. Window1_Create was written to take the phone number from the command line. We now want it to take the num-

```
1 call LtsNotes 'NOTESINIT', 'QUIET'
2 call LtsNotes 'DatabaseOpen','E:\NOTES\PCMAG.NSF','db_ID'
3 call LtsNotes 'DatabaseSearch', db_ID, 'writer', 'Select FORM = "Writer Info"'
4 do ThisDoc = 1 to writer.0
5   call LtsNotes 'DocRead', db_ID, writer.ThisDoc, 'docinfo'
6   do Field = 0 to docinfo.2.0
7     if docinfo.2.Field.1 = 'WriterName' then Person = docinfo.2.Field.3
8     if docinfo.2.Field.1 = 'Phone' then Leave
9     end
10  call VRMethod 'LB_1', 'AddString', Person '-' docinfo.2.Field.3
11  end
12 call LtsNotes 'databaseclose',db_ID
13 call LtsNotes 'notesterm'
```

Figure 5: **Here is the code to enter when creating the OS/2 version of the telephone dialer.**

```
Layout of the document information compound variable

Stem.1  Document ID
Stem.2.0  Number of fields (x) in this document
  Stem.2.x.1  Name of field x
  Stem.2.x.2  Data type (number, text, rich text, etc.)
  Stem.2.x.3  Field value
  Stem.2.x.4  Field options (protected, encrypted, name, signature, etc.)
  Stem.2.x.5  Field update command (used by DocWrite to update, add, or delete a field)
Stem.3  Date last modified
Stem.4  Document Class (Access control list, form, data, etc.)
Stem.5.y Document yes/no for privilege level y (y=1 to 5)
Stem.6  Unused
Stem.7  Origination ID information, used to control replication
```

Figure 6: **This is the layout of the *docinfo* compound variable.**

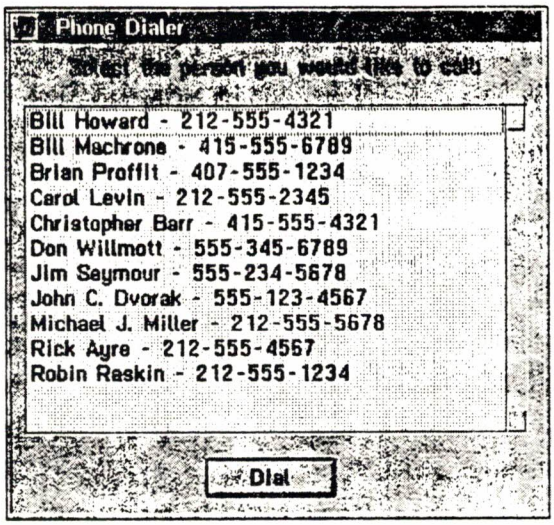


Figure 7: **This is the phone dialer produced under VX-REXX.**

ber as an argument, so change the line

```
Phone = Strip(InitArgs.1,, "'")
```

to read

```
Phone = Arg(1)
```

Also, since this is no longer the primary window, we need to load it manually with the instruction

```
DialWindow =
 VRLoad(VRWindow(),
 VRWindowPath(), Window2)
```

As in the Visual Basic version, we need to change the action taken when the user picks up the handset. Rather than ending the program, we want to close that window to return to the contact list, so replace the END instruction with

```
call VRDestroy DialWindow
```

The last bit of cleanup is to allow the user to double-click on a contact rather than clicking on the person and then the Dial button. Click on the list box with the right mouse button and select the DoubleClick event. Enter the following instruction:

```
Call PB_1_Click
```

The program will then handle double-clicks automatically. Save your project, create an executable, and directly attached to your Notes database you will now have the new automatic phone dialer shown in Figure 7.

ONLY THE BEGINNING Both VB/Link and InterfloX allow much more complex manipulation of Notes databases than I've demonstrated in this column. One reason people limit their use of Lotus Notes is that the data isn't sufficiently accessible outside Notes, as it would be when using a more traditional database. As you can see, however, the right tool can open up your Notes information to as many manipulations as you can imagine. □

BRIAN PROFFIT IS A SENIOR CONTRIBUTING EDITOR FOR *OS/2 MAGAZINE* AND *OS/2 DEVELOPER*, AND IS THE AUTHOR OF *OS/2 APPLICATION DEVELOPMENT TOOLS* (PREMIER PUBLISHING) *AND OS/2 INSIDE & OUT* (OSBORNE MCGRAW-HILL).

PRO 0803 Programming with Distributed Objects

Roger Sessions

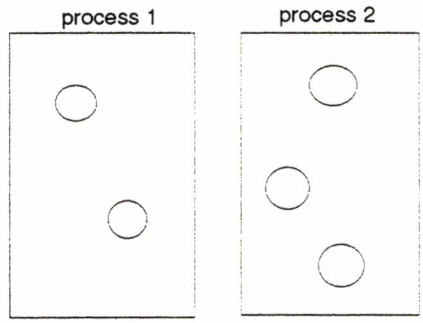
Copyright Notice

- This entire presentation is copyright by ObjectWatch, Inc., Austin, Texas. It may not be redistributed, in whole or in part, without permission.

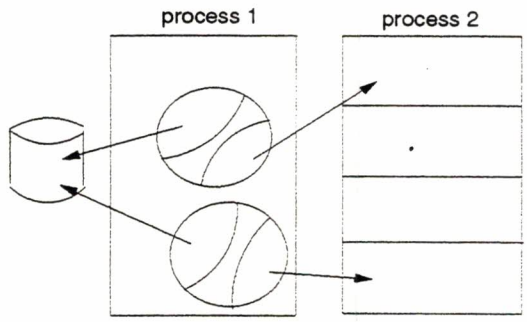
Overview of Talk

- Historical Background
- SOM Overview
- The pieces
- A simple Distributed SOM example

Objects in the 80s



Objects in the 90s



Objects in the 2000s

SOM

- The Leading Distributed Object Technology
  - Inheritance
  - Polymorphism
  - Encapsulation
  - Rich Object Model
- IBM's implementation of the OMG CORBA specification
- Committed to an open object distribution standard

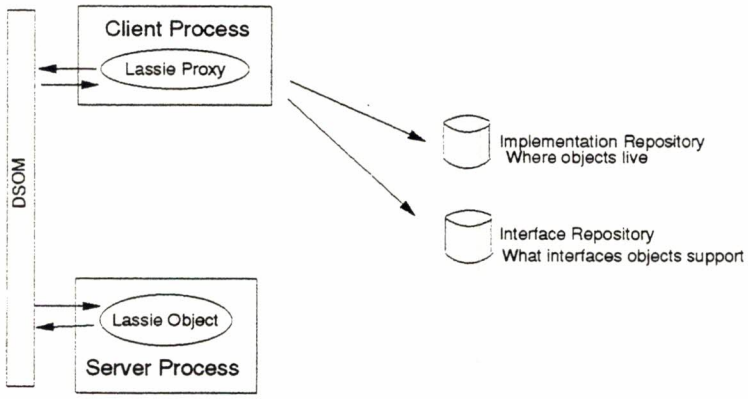
Scalability of Distribution

- Start with local objects
- Distribute objects in multiple processes
- Distribute objects across a network
- Distribute objects across a world
- ...

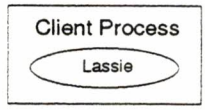
Networks Supported

- Single process (All)
- Shared memory (All)
- TCP/IP (AIX, OS/2, Windows)
- Netware IPX/SPX (AIX, OS/2, Windows)
- NetBIOS (OS/2)

SOM Architecture



User View of Architecture



The Programming Process

- Define an interface in IDL
- Use IDL Compiler to generate language bindings
- Implement interface in language of choice
- Build implementation as a DLL
- Write client program to access object using language bindings
  - Client programming language independent of implementation language

Defining the Interface

```
#include <somobj.idl>
interface dog : SOMObject {
  string getBark();
  string bark();
  string converse(in dog partner);

  implementation {
    releaseorder: getBark, bark, converse;
    dllname = "dog.dll";
  };

};
```

Implementing the Interface (1/x)

```
string  bark(dog somSelf,  Environment *ev)
{
   string myBarkString;
   string myBark;

   myBarkString = _getBark(somSelf, ev);
   myBark = (string) SOMMalloc(strlen(myBarkString) + 1);
   strcpy(myBark, myBarkString);
   return myBark;
}
```

Implementing the Interface (2/3)

```
string converse(dog somSelf,  Environment *ev, dog partner)
{
   string myBark, partnerBark, newString;

   myBark = _bark(somSelf, ev);
   partnerBark = _bark(partner, ev);

   newString = (string) SOMMalloc(strlen(myBark) +
               strlen(partnerBark) + 1);
   strcpy(newString, myBark);
   strcat(newString, partnerBark);

   return;
}
```

Implementing the Interface (3/3)

```
string  getBark(dog somSelf,  Environment *ev)
{
   static string myBarkString = "Unknown Dog Noise\n";
    return myBarkString;
}
```

Modifying Existing Interfaces - bigDog

```
#include <dog.idl>
interface bigDog : dog {
  implementation {
    override: getBark;
    dllname = "dog.dll";
  };
};
```

```
string getBark(bigDog somSelf,  Environment *ev)
{
   static string myBarkString = "Woof Woof\nWoof Woof\n\n";
   return myBarkString;
}
```

Modifying Existing Interfaces - littleDog

```
#include <dog.idl>
interface littleDog : dog {
  implementation {
    override: getBark;
    dllname = "dog.dll";
  };
};
```

```
string  getBark(littleDog somSelf,  Environment *ev)
{
   static string myBarkString = "woof woof\n";
  littleDogMethodDebug("littleDog","getBark");
   return myBarkString;
}
```

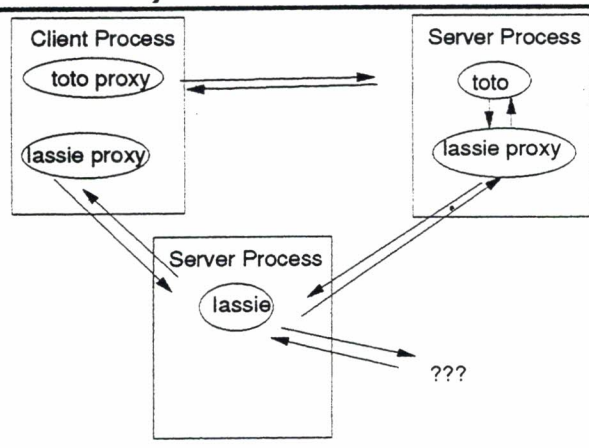
Client Code (1/2)

```
#include <somd.h>
#include <ldog.h>
#include <bdog.h>
#include <stdio.h>

void main()
{
 littleDog toto;
 bigDog lassie;
 string barkString;
 Environment *ev = SOM_CreateLocalEnvironment();

 SOMD_Init(ev);
```

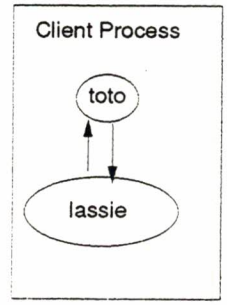
Client Code (2/2)

```
 toto = _somdNewObject(SOMD_ObjectMgr, ev, "littleDog", "");
 lassie = _somdNewObject(SOMD_ObjectMgr, ev, "bigDog", "");

 barkString = _converse(toto, ev, lassie);
 printf("%s", barkString);

 _somdDestroyObject(SOMD_ObjectMgr, ev, toto);
 _somdDestroyObject(SOMD_ObjectMgr, ev, lassie);

 SOMD_Uninit(ev);
 SOM_DestroyLocalEnvironment(ev);
}
```

Distributed Object Architecture



Client View



Creating a Distributed Object

- Creating an IDL definition
- Creating a language template
- Creating language bindings
- Add code to template
- Compiling the objects
- Creating a Dynamically Loaded Library
- Setting up the DSOM environment variables
- Updating the Interface Repository
- Choosing and registering a server
- Starting the somdd server
- Preparing a stub lib file for the dog code
- Compiling and linking the program
- Running the program

Creating an IDL definition

```
#include <dog.idl>
interface bigDog : dog {
  implementation {
    override: getBark;
    dllname = "dog.dll";
  };
};
```

Creating a language template

```
#ifndef SOM_Module_bdog_Source
#define SOM_Module_bdog_Source
#endif
#define bigDog_Class_Source

#include "bdog.ih"

SOM_Scope string  SOMLINK getBark(bigDog somSelf,
Environment *ev)
{
    bigDogMethodDebug("bigDog","getBark");
   return ;
}
```

Creating language bindings

```
sc -sh dog
sc -sih dog

sc -sh bdog
sc -sih dog

sc -sh ldog
sc -sih ldog
```

Add code to template

```
#ifndef SOM_Module_bdog_Source
#define SOM_Module_bdog_Source
#endif
#define bigDog_Class_Source

#include "bdog.ih"

SOM_Scope string  SOMLINK getBark(bigDog somSelf,
Environment *ev)
{
   static string myBarkString =
    "bigDog says \nWoof Woof\nWoof Woof\n\n";
   bigDogMethodDebug("bigDog","getBark");
   return myBarkString;
}
```

Compiling the objects

```
DLLOPTS = -D_OS2 -D_c -Q+ -GM+ -Ge- -Ti+ -Sm -c
INCLUDES = -I. -If:\som\include -If:\ibmcpp\include

 icc $(DLLOPTS) $(INCLUDES) dog.c
 icc $(DLLOPTS) $(INCLUDES) ldog.c
 icc $(DLLOPTS) $(INCLUDES) bdog.c
```

Creating a Dynamically Loaded Library

- Create Class Initialization Code (doginit.c)
- Create a DLL definition file (dog.def)
- Load the DLL

doginit.c

```
#include <dog.h>
#include <ldog.h>
#include <bdog.h>
#include <stdio.h>
#pragma linkage (SOMInitModule, system)

void SOMEXTERN SOMInitModule (integer4 majorVersion,
    integer4 minorVersion, string ignore)
{
  dogNewClass(0,0);
  littleDogNewClass(0,0);
  bigDogNewClass(0,0);
}
```

dog.def

```
LIBRARY dog INITINSTANCE
DESCRIPTION 'dog Class Library'
PROTMODE
DATA MULTIPLE NONSHARED LOADONCALL
EXPORTS
  dogCClassData
  dogClassData
  dogNewClass
  littleDogCClassData
  littleDogClassData
  littleDogNewClass
  bigDogCClassData
  bigDogClassData
  bigDogNewClass
  SOMInitModule
```

Loading the DLL

```
icc -Q+ -Gm+ -B"/exepack /packd /packc \
   /align:16 /pm:vio /MAP /NOI /NOL /DE" \
   -Fe dog.dll dog.obj bdog.obj ldog.obj \
   doginit.obj f:\som\lib\somtk.lib dog.def
```

Setting up the DSOM environment variables

```
set USER=roger
set HOSTNAME=mymachine
set SOMDDIR=f:\roger\testcode\t9
set SOMDDEBUG=1
set SOMIR=f:\roger\testcode\t9\som.ir
```

Updating the Interface Repository

```
sc -u dog
sc -u bdog
sc -u ldog
```

Choosing and registering a server

```
regimpl -A -i DogServer
regimpl -a -i DogServer -c bigDog
regimpl -a -i DogServer -c littleDog
regimpl -L -i DogServer
regimpl -l -i DogServer
```

Starting the somdd server

```
start somdd
```

Preparing a stub lib file for the dog code

```
implib /NOI dog.lib dog.def
```

Compiling and linking the program

```
EXOPTS = -D_OS2 -D_c -Q+ -Ge+ -W3 -Ti+ -Sm -Gm+
icc $(EXOPTS) -I. -Fe test1.exe \
      f:\som\lib\somtk.lib dog.lib test.c
```

Running the program

```
test1
```

Programs running

- somdd (Process responsible for starting up servers as needed
- somdserver - Process which will contain the dog objects
- test1.exe - Process which is accessing dog objects.

Other Nice SOM Features

- Binary independence
- Advanced Object Model
  - Classes represented as objects
  - Metaclass framework
- Compatible with existing languages
  - C++ Direct To Som Compilers

Summary

- Distributed Objects are the Future
- SOM Objects can be in another process or another country
- Fully compliant with the OMG CORBA specification for distributed object architectures
- An advanced object model

About the Author

Roger Sessions has spoken at dozens of conferences throughout the world and has written extensively about SOM and Object-Oriented Programming. He has given SOM tutorials at OOPSLA and other conferences and has consulted on many large scale SOM projects.

Roger was part of the IBM SOM development team from 1990-1995. He was the technical lead for the SOM Persistence Framework and a principal architect of the OMG's Persistent Object Service, an industry standard for Object Persistence based on the SOM Persistence Framework

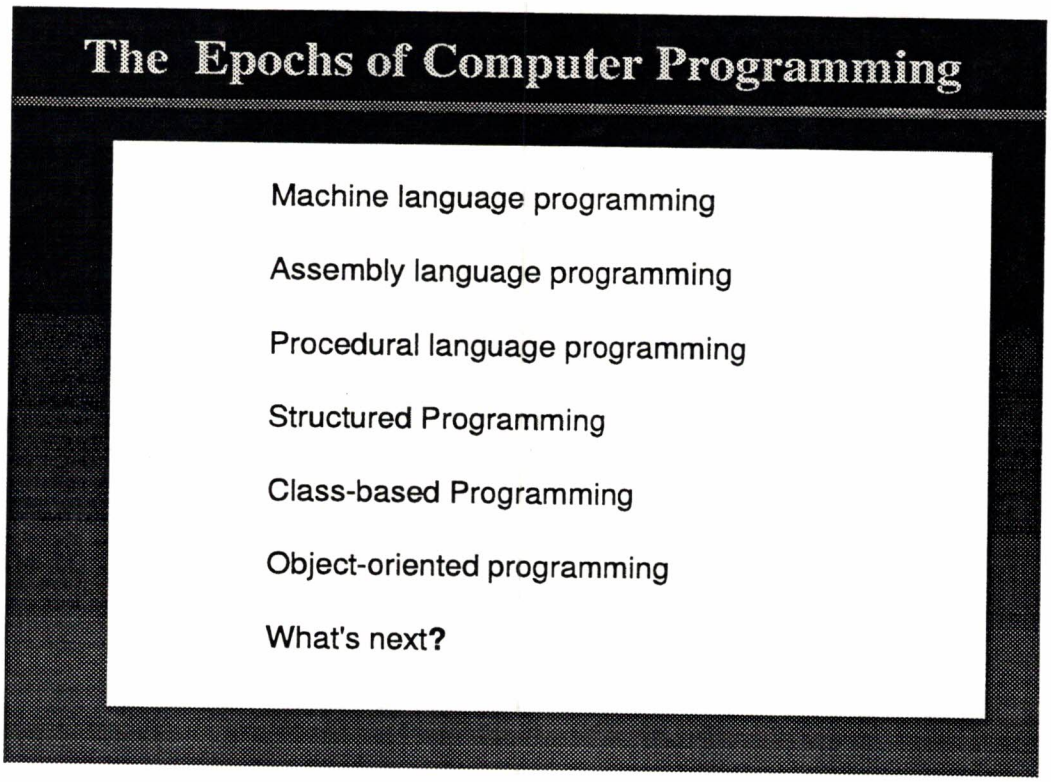
In 1995, Roger left IBM to form ObjectWatch, Inc., a company specializing in consulting, training, and mentoring in the use of SOM, DSOM, and other aspects of Distributed Object Technology.

Roger is the author of three books. His most recent book, scheduled for release in late 1995, is *Object Persistence: Beyond Object-Oriented Databases* , an upfront and personal view of the OMG Persistence Specification. He has also written *Class Construction in C and C++; Object-Oriented Fundamentals* and *Reusable Data Structures for C*. Roger has co-authored many articles in OS/2 publications.

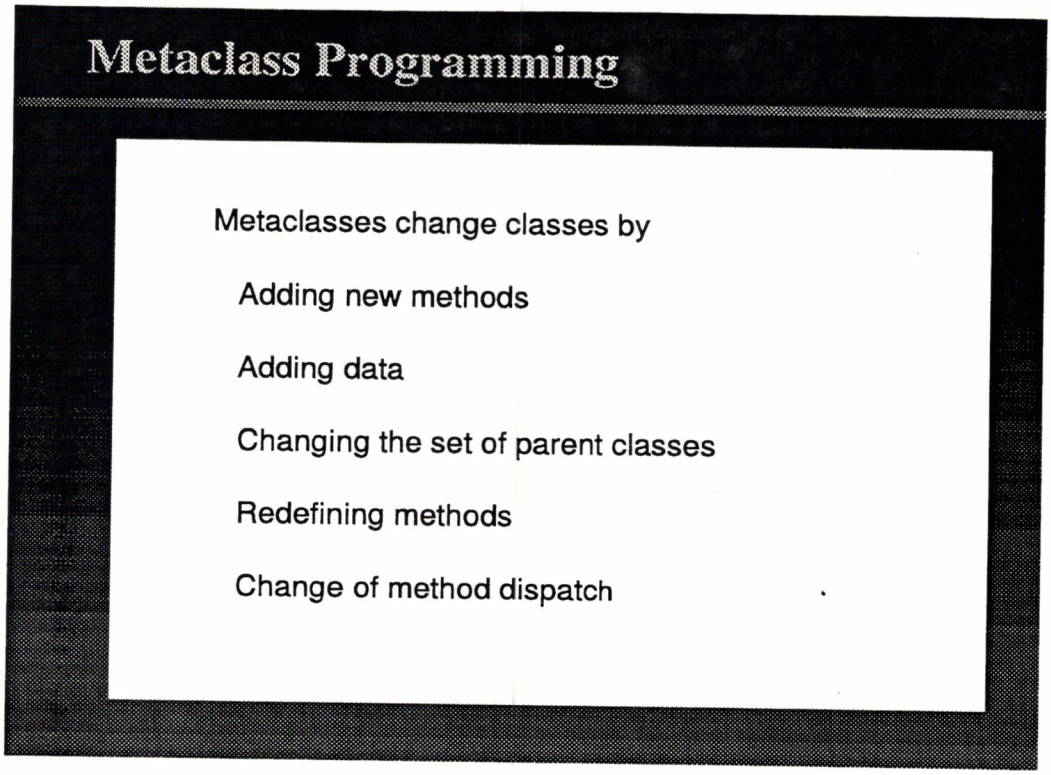
He can be reached at roger@fc.net.

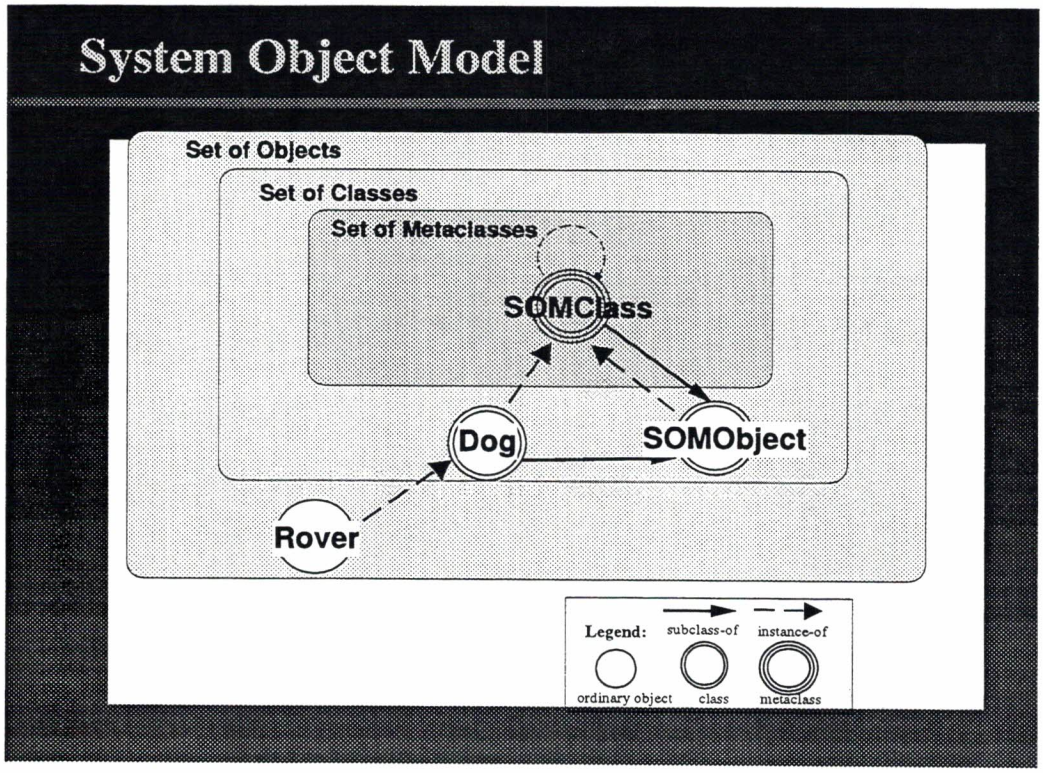
PRO 0805
SOM Metaclass Programming

Ira Forman
IBM

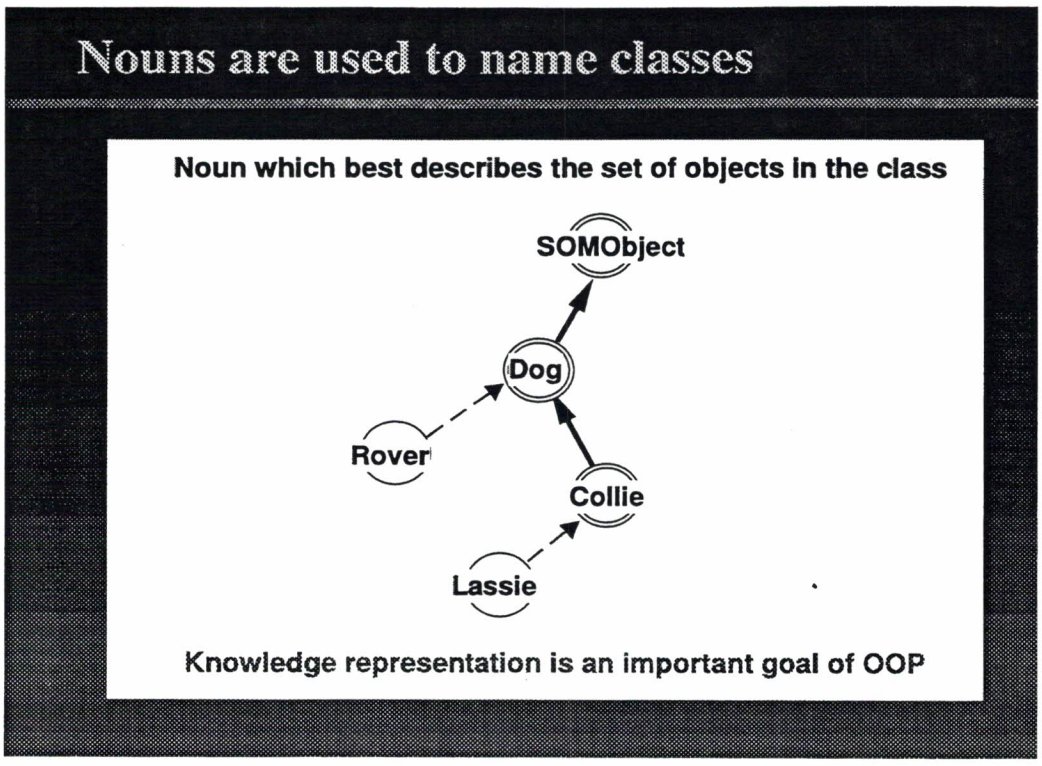


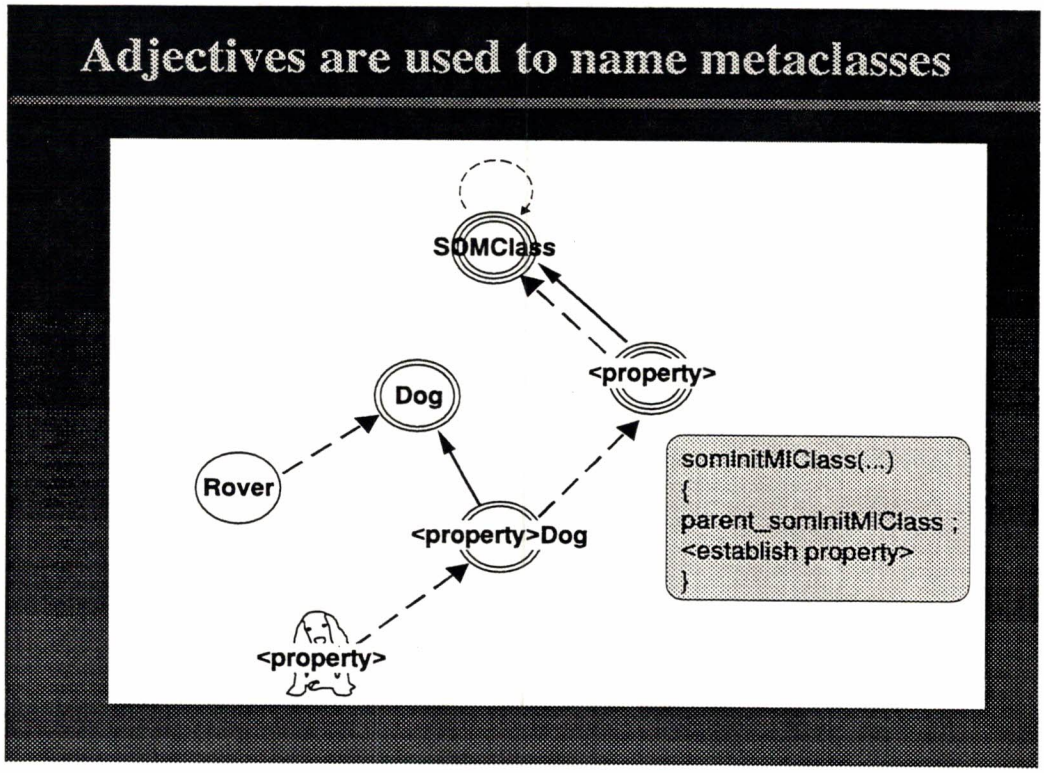
R0834A.PRS(LAN) 677J/003/IF



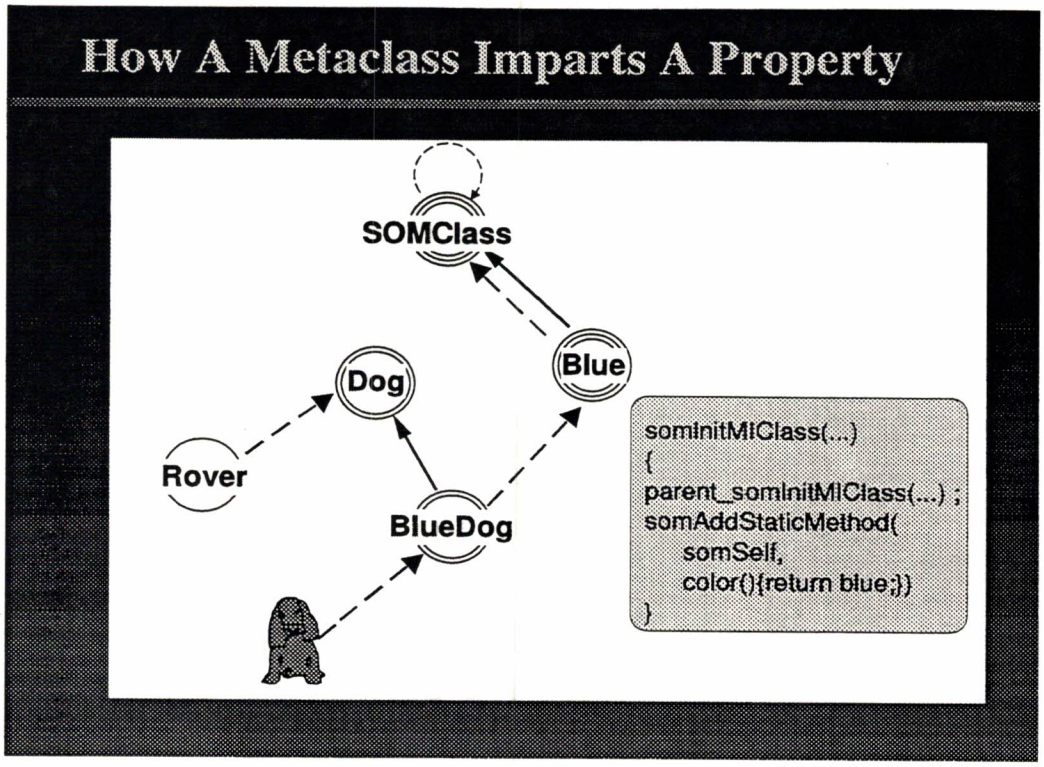


R0834A.PRS(LAN) 677J/003/IF



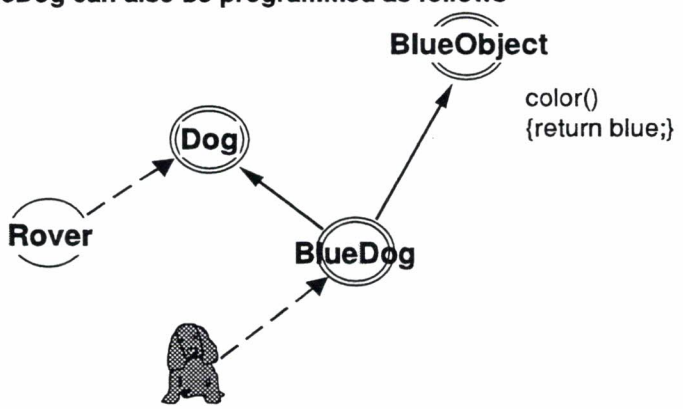


R0834A.PRS(LAN) 677J/003/IF



Yawn!!!

A BlueDog can also be programmed as follows

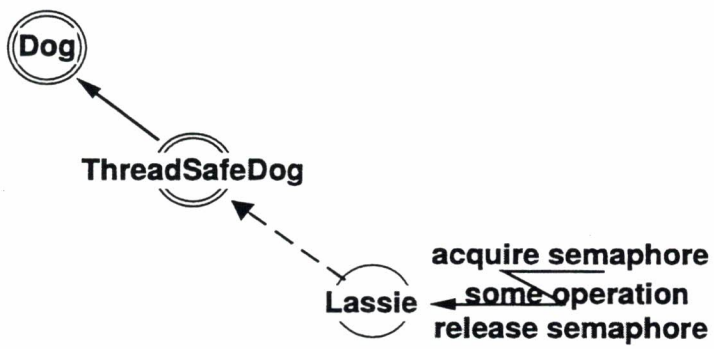


Because multiple inheritance is an operation on classes that creates a class with the union of the methods.

BUT...

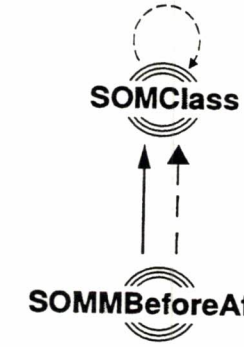
R0834A.PRS(LAN) 677J/003/IF

What about Thread Safety???



Acquiring and releasing semaphores are not just additional methods, rather thread safeness is an example of property that is wrapped **before** and **after** all methods.

The SOMMBeforeAfter Metaclass

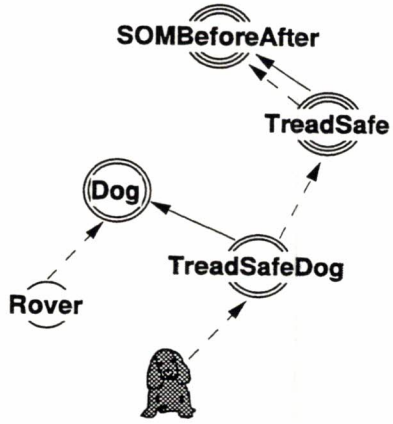


Create your own metaclass
1. Subclass SOMMBeforeAfter
2. Override sommBeforeMethod
3. Override sommAfterMethod

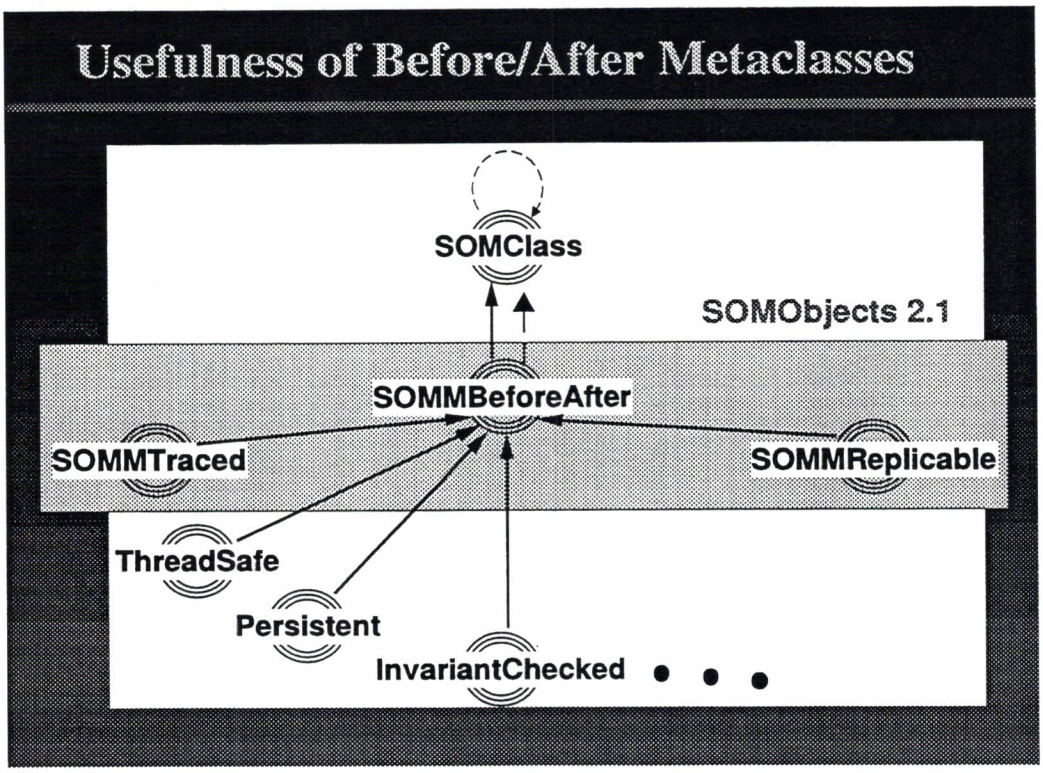
SOMMBeforeAfter takes care of the dispatching.

R0834A.PRS(LAN) 677J/003/IF

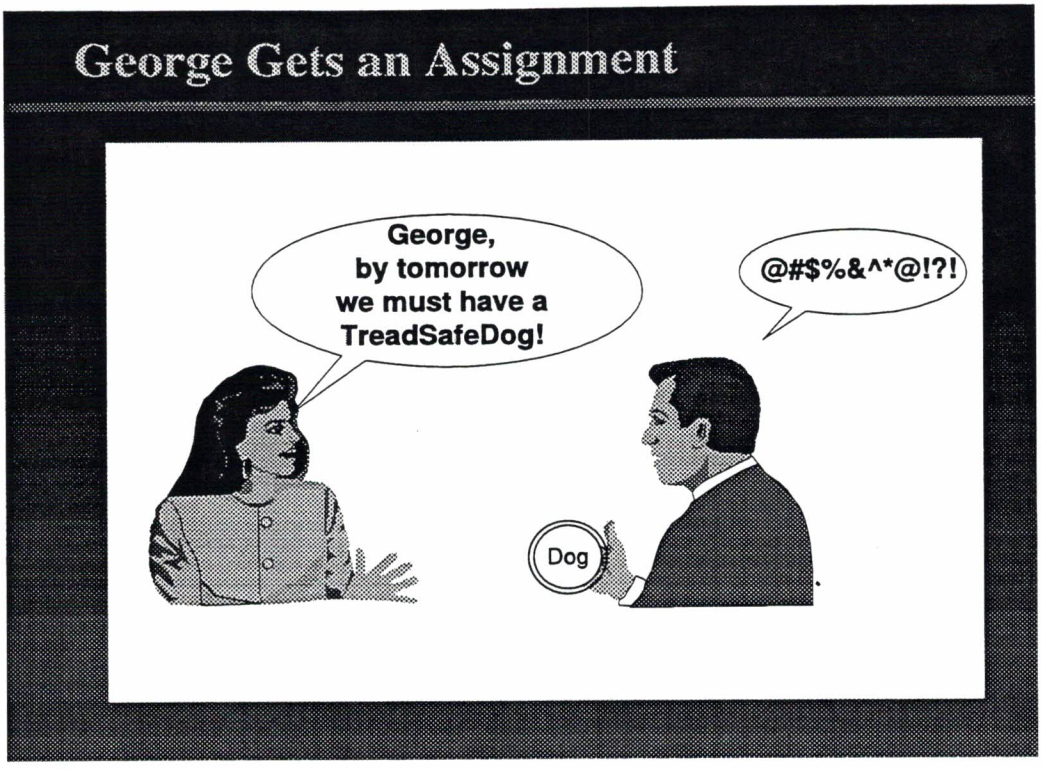
Example : ThreadSafeDog

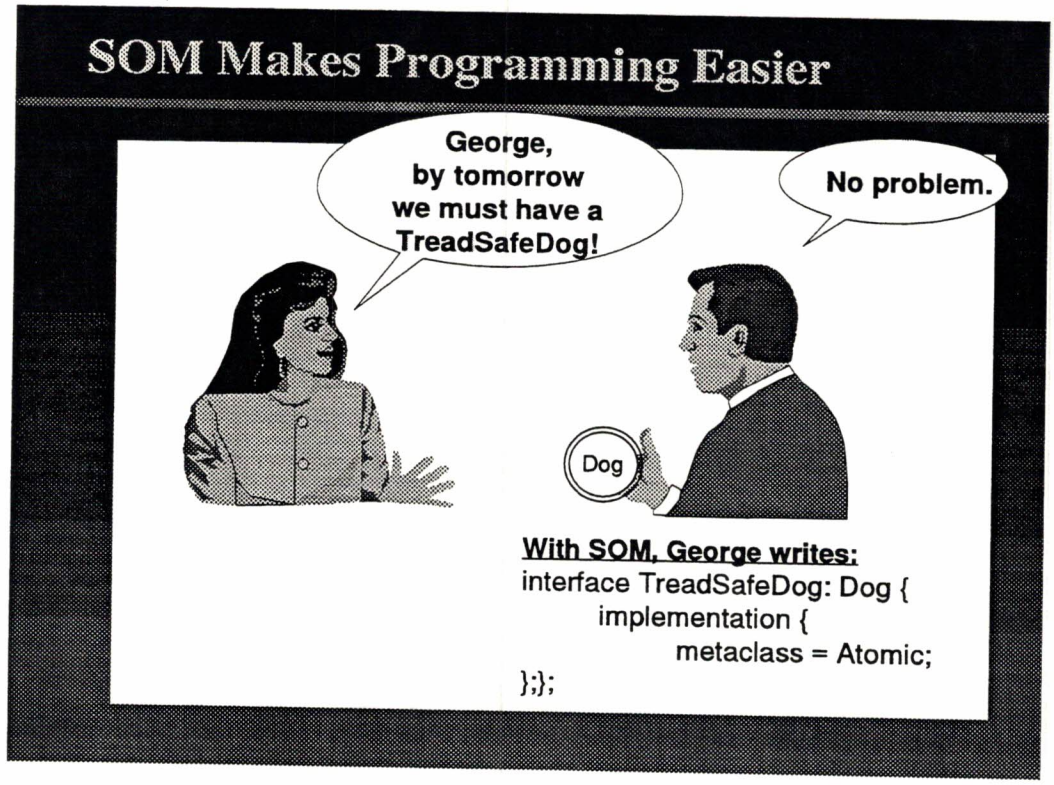


1. ThreadSafe is a subclass of SOMMBeforeAfter
2. sommBeforeMethod acquires a semaphore
3. sommAfterMethod releases a semaphore

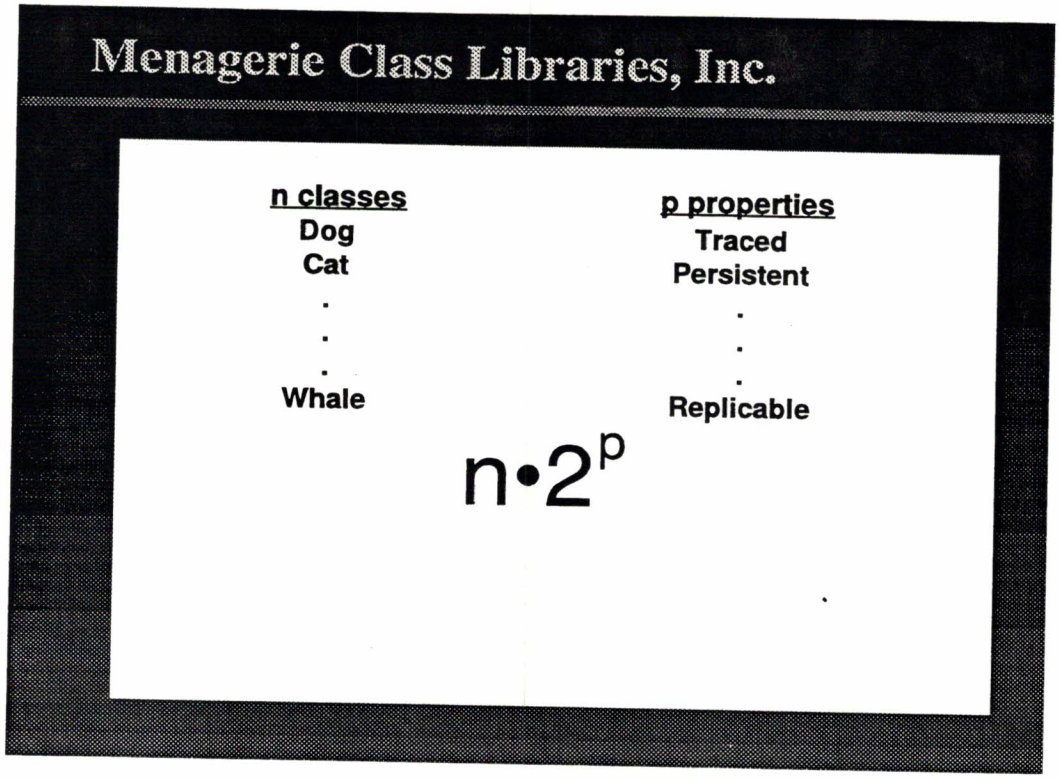


R0834A.PRS(LAN) 677J/003/IF





R0834A.PRS(LAN) 677J/003/IF



Smaller Class Libraries

n classes	p properties
Dog	Traced
Cat	Persistent
.	.
.	.
.	.
Whale	Replicable

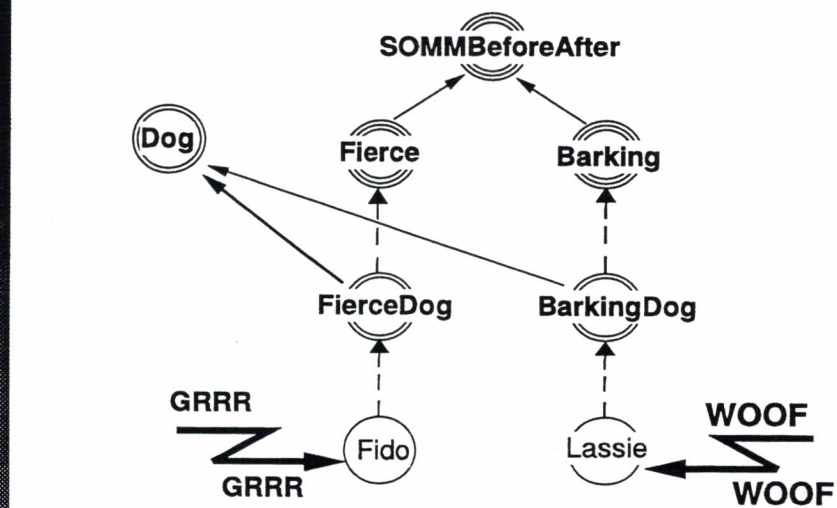
$$n \cdot 2^p$$

VS

$$n + p$$

R0834A.PRS(LAN) 677J/003/IF

Composition of Before/After Metaclasses



How is a Fierce Barking Dog programmed?

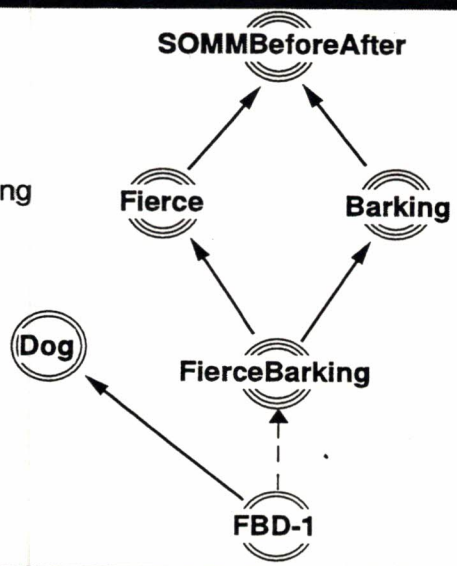
SOM IDL

```
interface <name>:<parent-list>{
    ...
    <method-declarations>
    ...
#ifdef___SOMIDL___
implementation{
    metaclass=<explicit-metaclass-name>;
    ...
};

#endif
};
```

R0834A.PRS(LAN) 677J/003/IF

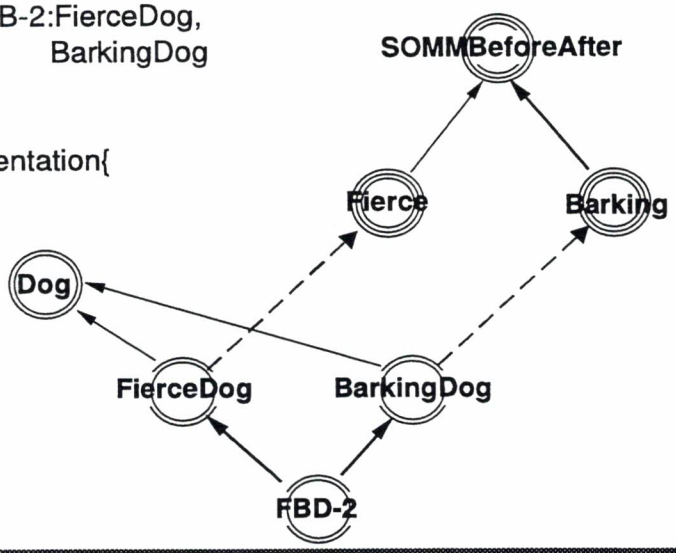
Technique 1

```
interface FB-1:Dog
{
    ...
    implementation{
      metaclass=FierceBarking
    };
};
```



Technique 2

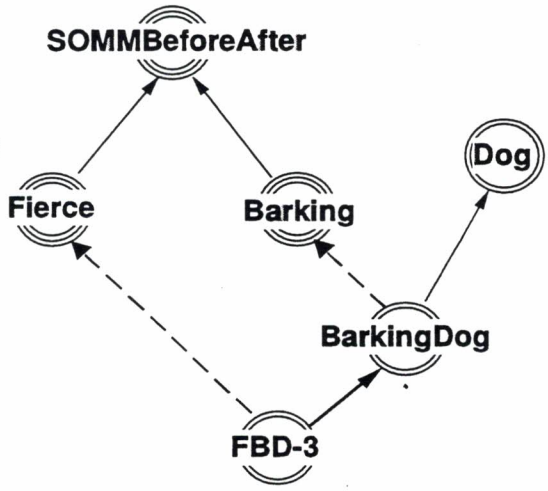
```
interface FB-2:FierceDog,
                BarkingDog
{
    ...
    implementation{
    };
};
```



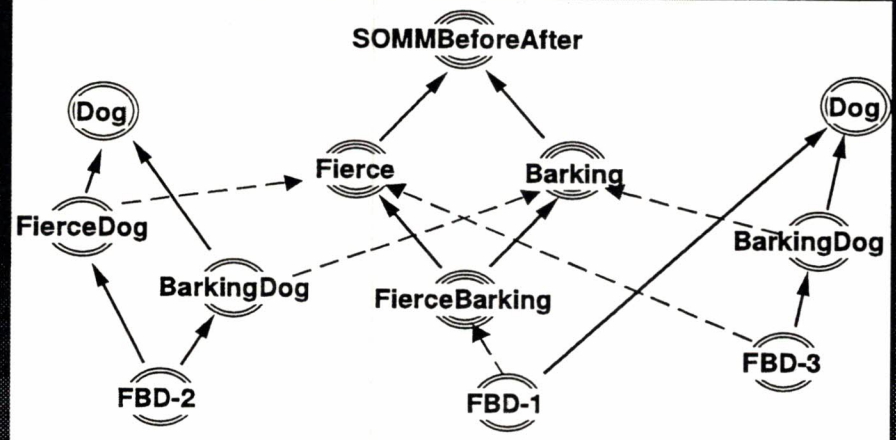
R0834A.PRS(LAN) 677J/003/IF

Technique 3

```
interface FBD-3:
BarkingDog
{
    ...
    implementation{
      metaclass=Fierce
    };
};
```



The Combined View



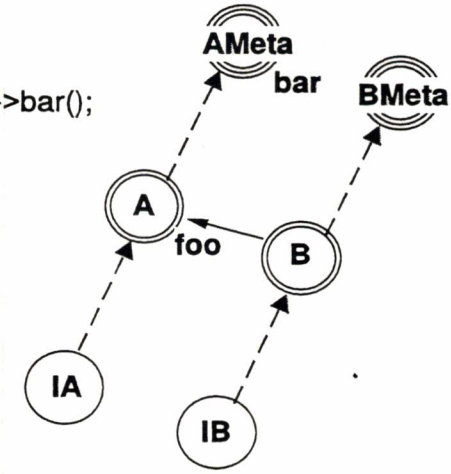
Should each of these method work: **Yes**
Should each have the same meaning: **Yes**
How: What do FBD-1, FBD-2, and FBD-3 have in common?

R0834A.PRS(LAN) 677J/003/IF

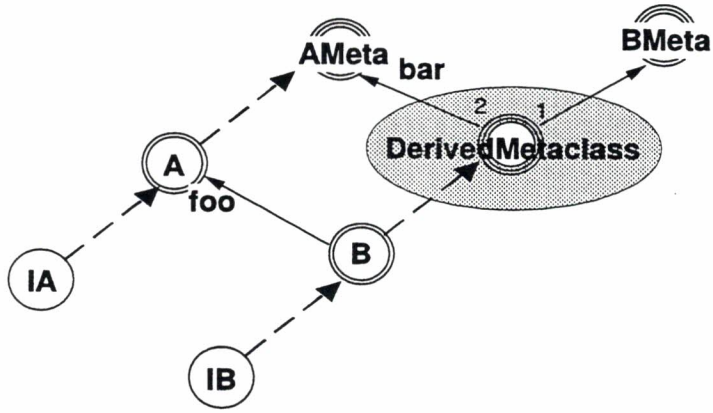
Metaclass Incompatibility

(does not occur in SOM)

```
interface A {
...
    void foo()
        {...
        self->somGetClass->bar();
        ...};
};

interface B:A {
...
    implementation{
        metaclass = BMeta
    ...
```



The Derived Metaclass

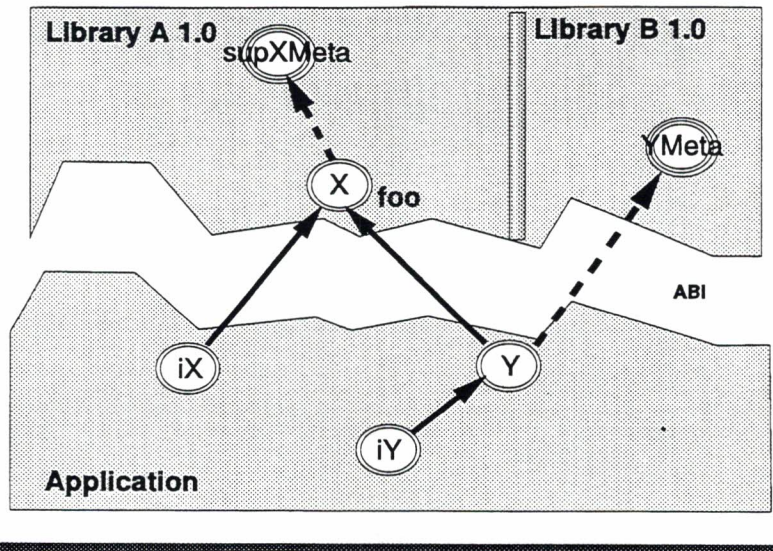


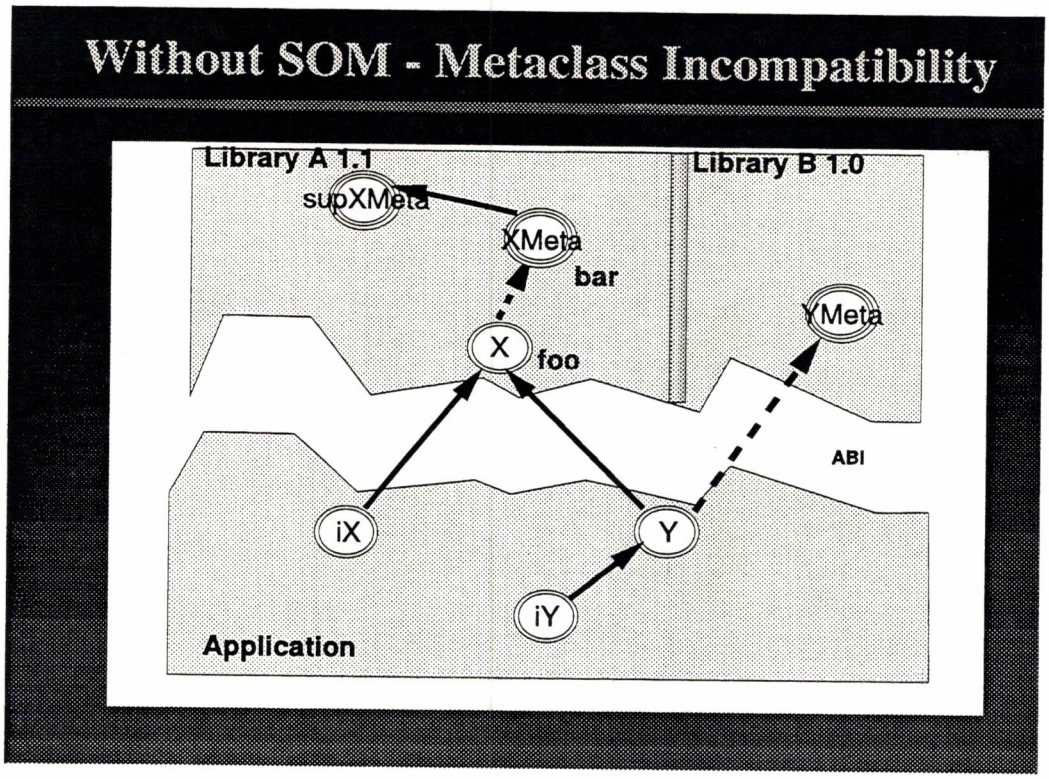
All object models solve the Metaclass Incompatibility Problem

Only SOM solves it without restricting the programmer

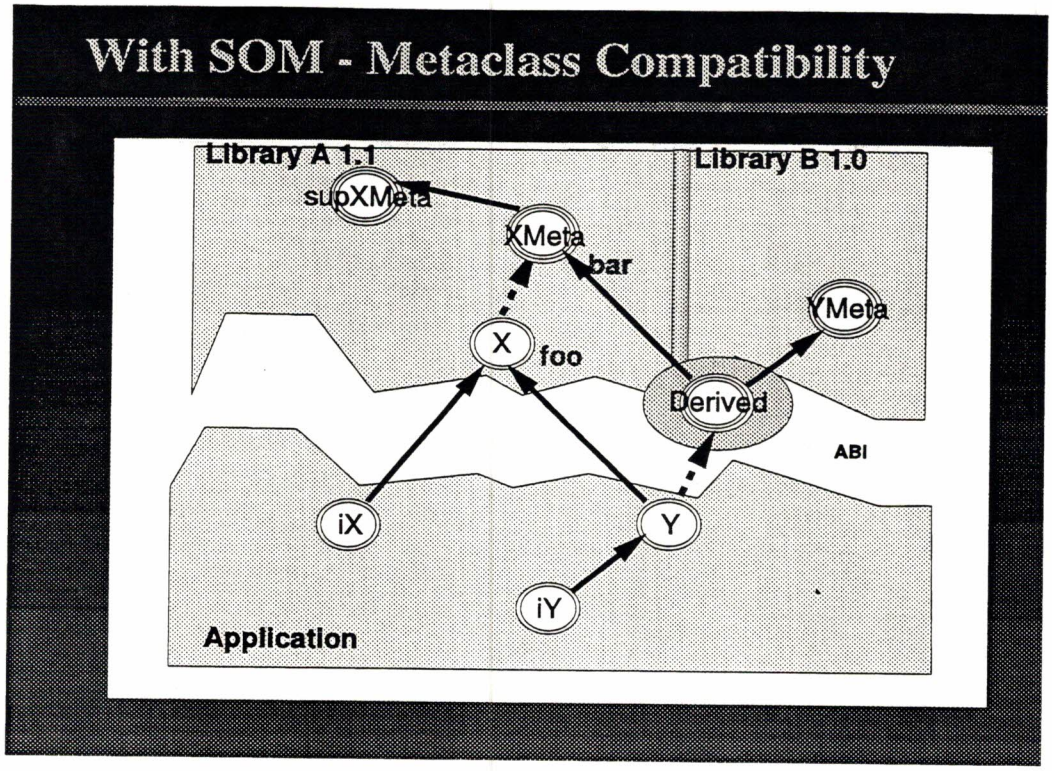
R0834A.PRS(LAN) 677J/003/IF

Consider

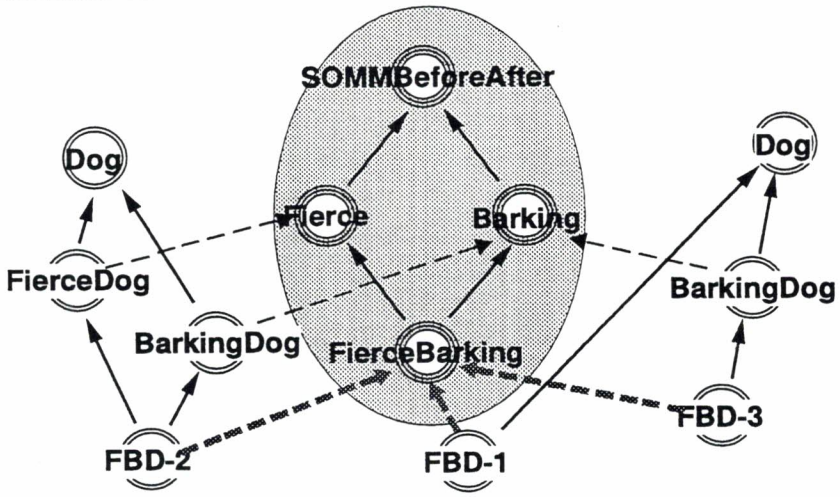




R0834A.PRS(LAN) 677J/003/IF



All have the same Derived Metaclass



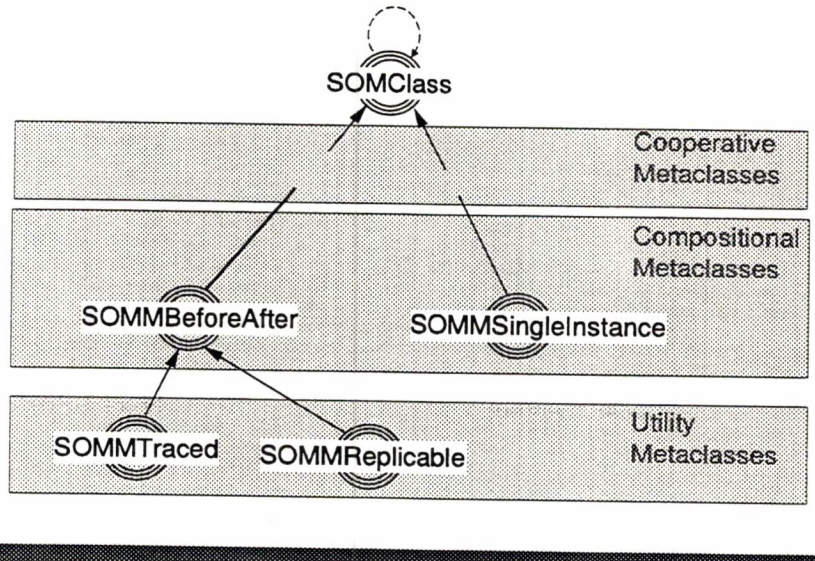
The metaclass hierarchy determines the composition

R0834A.PRS(LAN) 677J/003/IF

Dispatching Algorithm

```
BeforeAfterDispatch( simpleObject, ... )
    BeforMethodDispatch( ... )
    dispatch of primaryMethod
    AfterMethodDispatch( ... )

BeforeMethodDispatch( aMetaclass, simpleObject,
 ... )
    if aMetaclass defines BeforeMethod
        call that BeforeMethod
    else
        for all parents of aMetaclass
            call BeforeMethodDispatch( aParent, simpleObject, ... )

AfterMethodDispatch( aMetaclass, simpleObject, ... )
    if aMetaclass defines AfterMethod
        call that AfterMethod
    else
        for all parents of aMetaclass (in reverse order)
            call AfterMethodDispatch( aParent, simpleObject, ...)
```

SOMObjects 2.1 - Metaclass Framework



R0834A.PRS(LAN) 677J/003/IF

A Different (New?) View

Old Object-Oriented Programming (Wegner 1987)

OOP = objects + classes + inheritance

New Object-Oriented Programming (Forman & Danforth 1993)

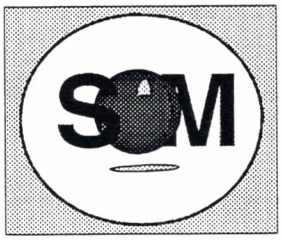
OOP = objects + classes + compositional operations

compositional operations = {
subclassing (i.e., extension, subalgebra, ...),
join (i.e., multiple inheritance),
meet (i.e., insertion of new classes)
substitution,
wrapper (i.e., before/after mertaclasses),
...
}

Conclusion - Program at a Higher Level

Metaclass Programming
enables the creation of
new operations on classes that
factor software in new ways

Making Reuse A Reality



R0834A.PRS(LAN) 677J/003/IF

PRO 0807
C Set ++ Class Library

Peter Devenyi
IBM

IBM Open Class Library - Topics

- C++ Class Library Strategy
- IBM Open Class Today
- Some Open Class Details
- Demos

Class Library/Framework Strategy:

IBM Open Class To Deliver...

Productivity - Reuse Code and Design

Abstraction - From low level complexities

Portability - Support Multiple Platforms

Function - Rich Class Libs and Framework

Evolution - into new Technologies

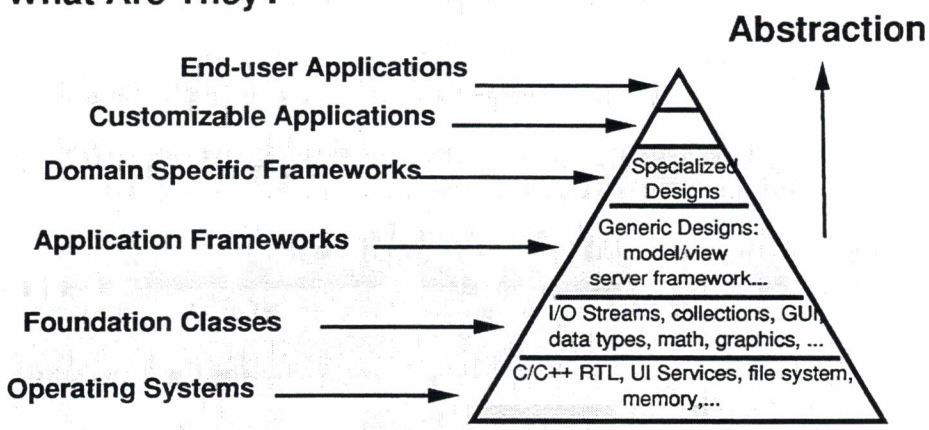
Class Libraries and Frameworks:

What are They?

- **Class Library**
  - Definition: "A collection of classes that implement standard program units which can be used as building blocks to create programs."
- **Framework**
  - Definition: "A framework consists of a collection of classes together a number of patterns of collaboration among instances of these classes."

Class Libraries and Frameworks:

What Are They?



Programming at Higher Levels of Abstraction Reduces Development Complexity

Class Library/Framework Strategy:

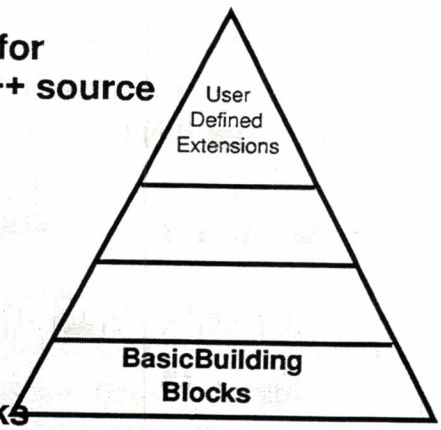
Class Library Coverage:

- Exploit range of IBM C++ Class Library Development Acti
  - Manage corporate-wide view of Class Library development and products
  - Package and ship widest possible range of C++ class libraries
  - Focus on common style guide, naming conventions and inheritence hiera

IBM Open Class - Strategy

Building Blocks

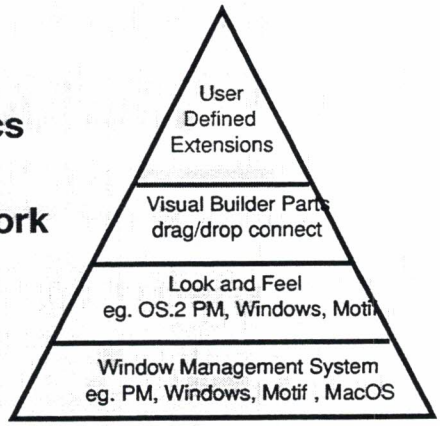
- Classes as the foundation for development of portable C++ source
  - Data types, collections
  - Input/output
  - Streaming
  - Exception, Notification, Messaging
  - Internationalization
- Support standards for C++
  - Class Libraries per ANSI/ISO draft
  - OMG, OSF, etc
- Base for classes, frameworks



IBM Open Class - Strategy

User Interface Classes

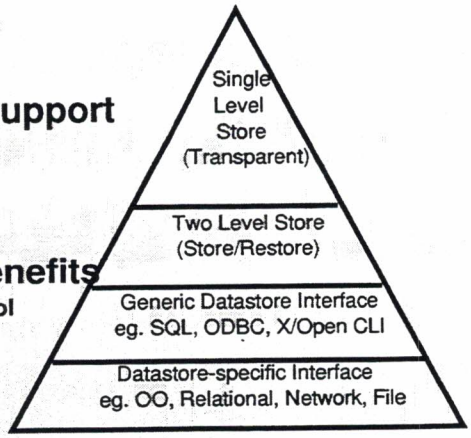
- Comprehensive GUI Support
- Insulate from Platform specifics
- Portable function
- Tightly integrated into framework



IBM Open Class - Strategy

Data Access

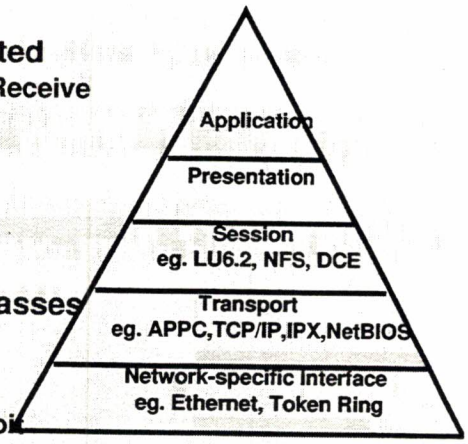
- Comprehensive Database Support
  - Four levels of Data Access
  - Efficiency and Flexibility
  - OMG Compliance
  - Rich set of Supporting Tools
- Significant Development Benefits
  - Tradeoffs between complexity and control
  - Portability across datastores



IBM Open Class - Strategy

Distribution:

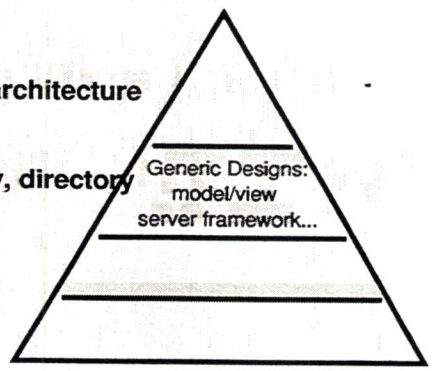
- Conversational Connection-Oriented
  - Initiate/Accept Conversations, Send/Receive
  - Native Support For SNA, TCP/IP, IPX
- Remote Procedure Call (RPC)
- Message Queues
  - Object-Oriented Interfaces
- Middleware-neutral distribution classes
  - for building distributed applications
- Distributed SOM
  - IBM Open Class Will Enable and Exploit



IBM Open Class - Strategy

Business Application Frameworks

- Compound Document
  - abstraction of underlying document architecture
- Distributed Application
  - abstractions of client, server, security, directory
- Framework based tools
  - generation of boiler-plate code

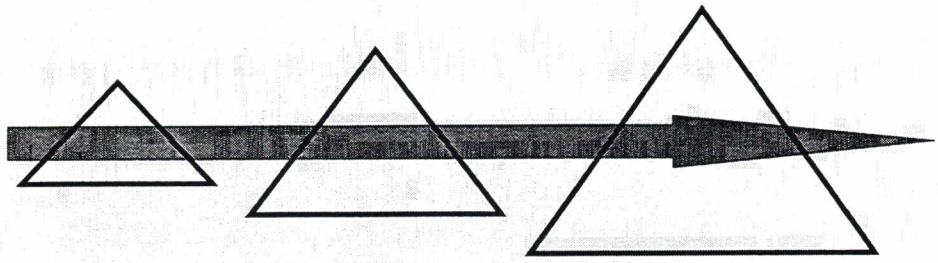


IBM Open Class and Taligent

Introduce Taligent Classes
- eg. Graphics, Print

Add More Taligent Classes
- eg. kernal services

Taligent/IBM CLASS Synergy
- IBM OpenDOC Framework Compati
  with Taligent Document Framew



Broaden IBM CLASS Coverage
- Data
- Communications

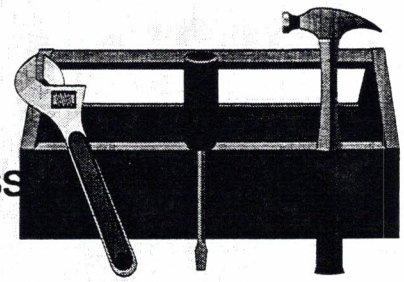
New IBM CLASS Content
- Data: More Databases
- Comm: CICS, MQI, DCE...

Support IBM CLASS Specializations
- eg. UI: Native Look and Feel

IBM Open CLASS Evolves Incorporating Taligent Techology !

IBM Open Class: Supporting Tools

- **Class Library and Framework Evolution**
  - Model/View Separation with abstractions for underlying document architec
  - Rich Set of Classes and for data access - anyhow and anywhere
  - Rich Set of Classes to support Distributed logic
- **VisualAge C++ "Assistants" Ease Work**
  - Visual Builder
  - Data Access Builder
  - Design Pattern Assistant
  - Framework Visualizer
  - Framework-Centric IDE
- **VisualAge C++ with IBM CLASS**
  - The Right Set Of Tools For The Job!



IBM Open Class - Platforms

- OS/2
- WINDOWS NT
- WINDOWS 95
- POWERMAC
- AIX
- HPUX
- SUN SOLARIS
- MVS
- OS/400
- ...

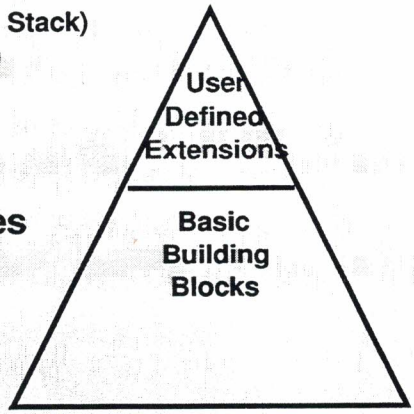
EVERYWHERE

IBM OPEN CLASS LIBRARY

IBM OPEN CLASS TODAY

IBM Open Class - Building Blocks

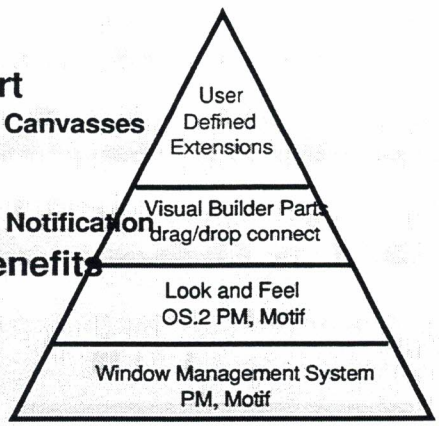
- **Collection Classes**
  - Sequences (Sequence, Queue(s), Stack)
  - Bags (Bag, Sorted, Ordered, Key)
  - Sets (Set, Sorted, Ordered, Key)
  - Maps, Relations and Trees
  - Cursor and Iterator
- **Application Support Classes**
  - String,Date, Time
  - Exception Handling
- **Standard Classes**
  - USL I/O Stream, Complex Math



IBM Open Class - UI Classes

User Interface Classes

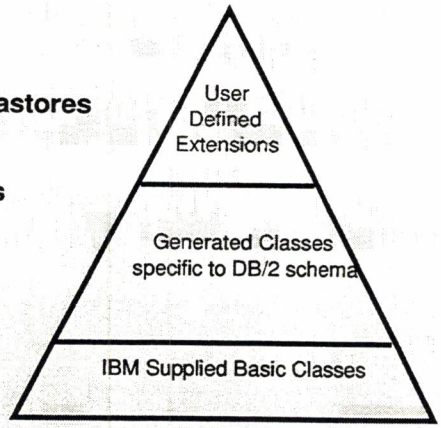
- **Comprehensive GUI Support**
  - Frame Windows, Control Classes , Canvasses
  - Events and Handler Classes
  - Menu, Drag/Drop, DDE
  - New: Toolbar, Clipboard, 2D , MM, Notification
- **Significant Development Benefits**
  - Increase in Productivity
  - CUA Look and Feel
  - Portable
  - Extendable



IBM Open Class - Data Access

OS/2 Database Access

- **Classes**
  - **Data Store Connection**
  - **Uniform Access to Multiple Datastores**
  - **Data Manipulation**
    - add, delete, update, retrieve
  - **Deal with Collections of Objects**
- **Complimentary Tools**
  - **Object Definer**
  - **Table Selection**
  - **Schema Mapper**
  - **Class Generation**
    - C++
    - IDL
    - Visual Builder Parts



IBM OPEN CLASS LIBRARY

SOME OPEN CLASS DETAILS

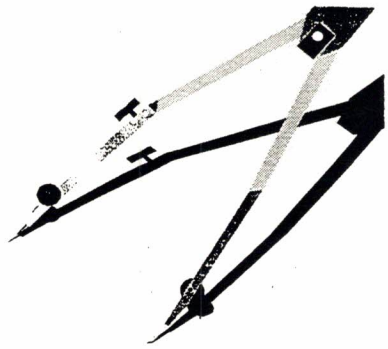
Components of the UI Library

- **Over 400 Classes**
  - 4000 member functions
- **Window Classes**
  - encapsulate basic building blocks for windows: title bars, buttons, menus, ..
- **Control Classes**
  - change the appearance or behaviour of windows
- **Menu, Notebook and Container Classes**
  - include menu bar, PopUp menu, sub-menu and system menu support
- **Event and Handler Classes**
  - event objects result from user or application action; handlers react to events
- **Application Classes**
  - control processes, threads and resources associated with the application
- **Data Type Classes**
  - model basic graphical data types such as points and rectangles

this page intentionally left blank

SYS 0804
Mastering NT Server

Mark Minasi
Mark Minasi & Company
Arlington, VA



Managing NT Server

a whole new network architecture

Presented by Mark Minasi
MMCO, inc.
5229 Wilson Blvd
Arlington, VA 22205
(703) 276-8940
(703) 276-1014 fax

What's in NT for you?

- Architecture independence
- Multiprocessor support
- C2 level security
- Networking based on TCP/IP, IPX/SPX, and NetBEUI
- Reasonably priced server
- Big memory model
- New file system
- Very reliable
- Variable levels of compatibility with Novell, TCP/IP, Macintosh
- True multitasking
- Supports remote access
- Available in a workstation or server format

NT security and administration

- Domain-based administration
- Single logon allows access to all authorized domains and resources
- Complete auditing information available
- Built-in performance monitor

NT problems

- NT really prefers NT and Windows for Workgroups clients.
- You can only do administration from WfW or NT workstations, or at the server.
- The TCP/IP integration is not complete; the naming service is not DNS (the Internet standard), but rather a Microsoft system called WINS. Also, there is no NFS client shipped.
- Security model slows down existing Windows programs.
- No Banyan integration to date.
- No fax server shipped with NT, and modem sharing can only occur with NT workstations.
- Impossible to set disk quotas.
- The backup program is a bit limited, but there are third party alternatives.

What do you need to run NT?

- 486DX-33 or faster
- 70 MB disk space
- 16 MB RAM
- CD-ROM and SCSI strongly recommended
- PCI, EISA, or Micro Channel buses are a good idea due to NT's multitasking nature.

Is NT completed?

- Definitely not.
- Still sports Windows 3.x look, although not (thankfully) its GP faults and instability.
- May soon see an NT 3.6 with PCMCIA support and the Win95 front end.
- NT 4.0, or NT 96 ("Cairo") will be object oriented, have Win95 front end, and will be a complete re-design.

NT and Novell

- NT ships with a NetWare client, making it possible for any NT machine to use NetWare services.
- An NT/Novell gateway makes it possible for people to access Novell resources through NT — with the side effect of bypassing Novell connection limits.
- There is, however, no way to link Novell user accounts and NT user accounts, requiring dual administration.
- A NetWare migration tool eases the job of initial account creation.

NT and the iNTernet

- True multitasking makes NT a good platform for ftp and http servers. ("On the Internet, no one knows you're running NT.")
- NT ships with an ftp server, and a public domain http server is around.
- The Dynamic Host Configuration Protocol (DHCP) saves you the trouble of keeping track of IP addresses.
- NT machines can serve as IP routers, albeit inefficiently.
- NT includes the important dial-up Internet protocols SLIP and PPP

SYS 0806
Building Successful DB2/2 Applications

Mike Logan
IBM

Agenda

- General Design Concepts
- Cursors, Locks, & Isolation
- Extended SQL
- Designing for Performance
- Precompiler Considerations
- Miscellaneous Hints & Tips

General Design Concepts

Agenda:

- Connection Strategy
- Multi-Threading
- Error Handling
- Interrupts

Connection Strategies

- Conversational Design
  - Little performance penalty
  - Avoids multiple connect/resets
- Pseudo-Conversational Design
  - Reduces server memory usage
  - Big performance penalty
  - Can be used for multi-database access

Connect Functions

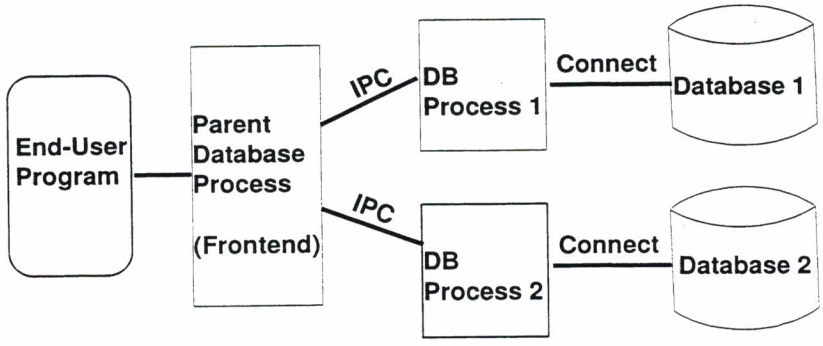
- A Connect performs:
  - Client:
    - Allocates Communications & Remote Services Heaps
    - Allocates conversation
  - Server:
    - Allocates Communications & Remote Services Heaps
    - Starts background agent
    - May allocate database caches
    - Validate Userid/Password

Connect Rules of Thumb

- Connect should be called from a separate thread
  - Violates 1/10 second rule
- Can hang application
- During application initialization if possible

Handling Multiple Connections

Preferred Method in Version 1.x
Multi-Process Implementation



Handling Multiple Connections

Preferred Method in Version 2.1
Distributed Unit of Work (DUOW)

Assuming Precompile option CONNECT 2:

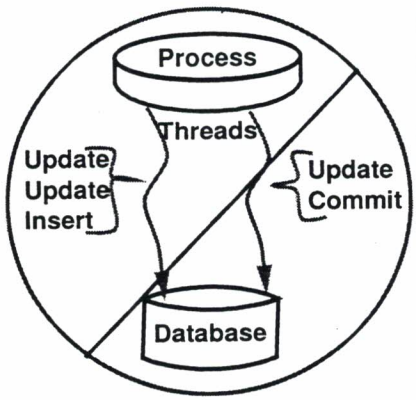
```
EXEC SQL CONNECT TO DATABASE1;
EXEC SQL SELECT * FROM TABLEA;
...
EXEC SQL CONNECT TO DATABASE 2;
EXEC SQL SELECT * FROM TABLEB;
...
EXEC SQL SET CONNECTION DATABASE1;
EXEC SQL UPDATE TABLEA SET ...;
```

Multi-Threaded Engine

- Version 1.x implemented a multi-process engine
  - Each connection requires a separate OS/2 process
- Version 2.1 implements a multi-threaded engine
- Each connection requires a separate OS/2 thread
  - Reduced overhead
    - Improved connection time
    - Reduced memory requirements per connection
  - Does not change how threads are treated at the client

Multi-Threaded Applications

DB2/2



- UOW at process level
- All threads committed when any thread committed
- All access to the database within a process is serialized

Error Handling

DB2/2

- Don't use WHENEVER sqlerror
  - Must have an error label for each SQL function
  - Uses 'C' labels
  - Have no control for error recovery
- Use common error routine
  - Usually placed in a DLL
  - Common error checking for all applications
  - Provides for error recovery

Interrupts

- Interrupt API
  - Uses
    - Stop long running query
    - Abort undesired transaction
    - Lock timeout for Version 1.x
  - Must be called from separate thread
  - Can use WM_TIMER message or semaphore as timer

Cursors, Locks, & Isolation

Agenda:

- Cursors
- Isolation Levels
- Locking
- Deadlocks

Cursors: Definition & Usage

- Programatic way to retrieve a set of rows from a database
  - Declare Cursor
  - Open Cursor
  - Fetch Cursor
  - Close Cursor
- Cursor remains open until a Commit or Close Cursor
  - With Hold option keeps cursor across multiple UOW
- No DB2/2 defined limit on # of cursors
- Cursor names must be unique within .sqc, .sqx

Cursor Categories

- Three categories
  - Read Only
  - Updateable
  - Ambiguous
    - Dynamic cursors are always ambiguous
- Row blocking influenced by type of cursor

Isolation Levels

- Degree of isolation from other conncurrent SQL transactions
  - Determines the type and duration of most locks
- 4 types
  - Repeatible Read
  - Read Stability - Version 2.1
  - Cursor Stability
  - Uncommitted Read

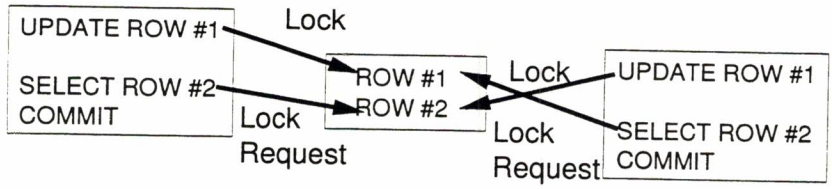
Lock Compatibility Matrix

	State of Held Resource							
State Being Requested	IN	IS	S	IX	SIX	U	X	Z
IN	yes	yes	yes	yes	yes	yes	yes	no
IS	yes	yes	yes	yes	yes	yes	no	no
S	yes	yes	yes	no	no	yes	no	no
IX	yes	yes	no	yes	no	no	no	no
SIX	yes	yes	no	no	no	no	no	no
U	yes	yes	yes	no	no	no	no	no
X	yes	no	no	no	no	no	no	no
Z	no	no	no	no	no	no	no	no

IN = Intent None
IS = Intent Share
S = Share
IX = Intent Exclusive
SIX = Share Intent Exclusive
U = Update
X = Exclusive
Z = Super Exclusive

Deadlocks (-911)

- Two programs, each holding resources the other needs
- Deadlock detector runs every DLCHKTIME milliseconds
  - Selects one of the two and rolls it back with -911
  - The other proceeds normally
- Application should attempt to retry the transaction



Extended SQL

Agenda:

- Version 2.1 Extensions
- User Defined Types
- User Defined Functions
- Triggers

User Defined Types

- Non-standard types based on built in types
  - Same internal representation
  - New type and base type are distinct
    - Only functions defined for new type may be applied
    - Generates comparison and casting functions
- Provide data abstraction
- Flexibility
  - You control semantics and behavior of the new type
- Performance
  - Can be optimized as efficiently as base type

```
CREATE DISTINCT TYPE AUDIO AS BLOB(75K)
```

User Defined Functions

- Allows creation of scalar functions
  - May operate on built-in or UDT's
  - Written in 3GL
  - Cannot contain SQL
  - Can be invoked from SQL, triggers, or view definition
  - Name is case sensitive
- Two types
  - Fenced - separated from database engine
  - Not Fenced - runs as part of engine
    - Faster
    - Can bring down the entire engine

Triggers

- User defined set of actions that occur before or after INSERT, DELETE, or UPDATE
- Possible uses
  - Validate input data
  - Enforce business rules
  - Create an audit-trail
  - Almost anything when used with UDF's
- Benefits
  - Faster application development
  - Easier maintenance
  - Global scope

Designing for Performance

Agenda:

- Static vs. Dynamic SQL
- Compound SQL
- Stored Procedures

Static vs. Dynamic SQL

- Static SQL
  - SQL compiled before runtime
  - Usually outperforms dynamic
  - A requirement for OLTP transactions
- Dynamic SQL
  - Compiled at runtime
  - Suited only for ad-hoc queries
  - Can outperform static in some circumstances

Compound SQL

- Version 2.1 only
- Multiple SQL statements in a single executable block
  - Reduces DB2/2 and network overhead
  - Static SQL only
- Two types
  - Atomic
    - If any statement fails, all are rolled back
  - Not Atomic
    - Failure of one statement does not affect others
    - Only type supported by DDCS/2

Stored Procedures

- Code stored on server
- Executes on server
- Invoked by client
- Implementation
  - DLL (C, Cobol)
  - CMD (Rexx)
- Very flexible
  - Not limited to database functions
  - Also may be run as Fenced or Not Fenced

Uses for Stored Procedures

- Typically, tasks that require a lot of work but return little data back to the client
  - Transaction Processing
  - Batch processing
  - Data manipulation not easily accomplished with SQL
- Access to resources on DB2/2 server
  - Through DosExecPgm
  - API's only supported locally
- Serialization of access to server resources

Stored Procedure Invocation

- Order of operations
  - EXEC SQL CONNECT
  - Allocate sqlda and sqlchar
  - allocate sqlvar and sqlind elements
  - sqleproc(...) *CALL in Version 2.1
  - EXEC SQL CONNECT RESET

Benefits of Stored Procedures

- Performance
- Reduced network traffic
- Co-operative processing
- Ease of application maintenance
  - Single point of change
- Can take advantage of greater server resources

Disadvantages of Stored Procedures

- Complex implementation
- Not maintained as part of the database
  - Not included in a database backup
- Increased workload on server

Precomplier Consideration

Agenda:

- C++ toleration
- Limitations

C++ Toleration

- All C++ constructs, key words, etc. permitted
  - Outside of SQL statements
- const/volatile modifiers allowed but ignored
- Class data members may be host variables
  - Not classes or objects themselves
- SQL statements may be placed anywhere

Limitations

- Host variables must be unique within module
- All header files and generated code will be C
- No SQL classes provided
- No '.', '->', '::' permitted in host variable references

Miscellaneous Hints & Tips

- Dealing with LOB's
  - Careful where you allocate them
- Embedding Host Variables or SQL in class definition
  - Requires all modules that include it to be precompiled

SYS 0808
Network Security and Single Signon

Scott Baumann
IBM

NetSP Matrix

PLATFORM	Release	(GSS-API) Client	(GSS-API) Server	(User Interface) GUI	(User Interface) Cmd line	Conn. to A.S.	3270** Emul
OS/2	2.1	✓	✓	✓	✓	TCP/IP ✓ NetBios ✓ LU 6.2 ✓ IPX ✓	CM/2 ✓ Rumba ✓
DOS	DOS 3.3 MS-DOS 6.0	✓	N/A	N/A	✓	NetBios ✓ IPX ✓	PC3270 ✓ Irma DCA ✓ Attachmate ✓
DOS/Windows	Windows 3.1	✓	✓	✓	N/A	TCP/IP ✓ NetBios ✓ IPX ✓	PC3270 ✓ Irma DCA ✓ Attachmate ✓ Rumba ✓
AIX/6000	3.2	✓	✓	✓	✓	TCP/IP ✓ NetBios ✓ LU 6.2 ✓	X3270 ✓
SUN	1.1	✓	✓		✓	TCP/IP ✓	X3270 ✓
HP-UX	V9.0	✓	✓	✓	✓	TCP/IP ✓	X3270 ✓

** 3270 Emulator provides single secure signon from the client workstation to RACF 3270 applications using the PassTicket technology

N/A = not applicable
✓ = already shipped in GA Product
* = Beta code available/to be available
Releases indicate minimum requirements

Single Signon Matrix

One Password, One Userid

TO:	FROM: OS/2	DOS	DOS/W	AIX/6000	SUN	HP
RACF MVS/VM 1.9.2	CM/2 ✓ Rumba ✓ Get_PassTicket ✓	PC3270 ✓ Irma DCA ✓ Attachmate ✓ Get_PassTicket ✓	PC3270 ✓ Irma DCA ✓ Attachmate ✓ Rumba ✓ Get_PassTicket ✓	X3270 ✓ Get_PassTicket ✓	X3270 ✓ Get_PassTicket ✓	X3270 ✓ Get_PassTicket ✓
OS/2 Telnet Server 2.0	✓	N/A	N/A	N/A	N/A	N/A
OS/2 LAN Server 2.0	✓	✓	✓	N/A	N/A	N/A
Novell 3.1.2, 4.01	✓	✓	✓	N/A	N/A	N/A
GSS-API enabled Applications	✓	✓	✓	✓	✓	✓

✓ = already shipped in GA Product
* = Beta code available/to be available
Releases indicate minimum requirements

NetSP Secured Logon Coordinator V1R2

April 24, 1995

Scott Baumann
Technical Support

Network Security Products
RTP, NC

(919) 254-7416	Voice
(919) 254-4289	Fax
sbaumann@vnet.ibm.com	Internet
USIB5QFK@IBMMail	IBMMail

Trademarks

The following terms are used in this presentation and are the trademarks of the IBM Corporation:

IBM	TSO
AIX/6000	CICS
OS/400	IMS
OS/2	CMS
DOS	

The following terms are used in this presentation and are the trademarks of other companies:

UNIX - UNIX Systems Laboratories	IPX - Novell
DEC - Digital Equipment Corporation	HP - Hewlett Packard Co.
SUN - SUN Microsystems Inc	DCE - Open Software Foundation
Novell - Novell	

Network Security is a SERIOUS customer problem!

Wall Street Journal (6/7/93):

- $864M loss due to credit card fraud last year

Computer Emergency Response Team:

- 5-7 incidents/day involving 5-6 users
- One incident involved about 62,000 users (Internet)

Every night $30 Trillion transferred between banks & institutions

Survey Finds Losses Linked to Lax Computer Security

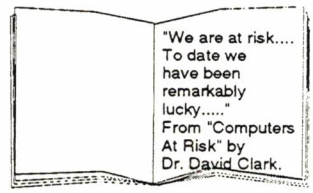
"More than half of the companies in a new us survey say they have had financial losses due to lax computer security.

The new study, based on interviews Ernst & Young conducted with 1,271 companies of various sizes, found that more than half admitted to having sustained losses, sometimes of more than 1 million dollars, due to security problems."

WSJ 22.11.94 p.4

Computerworld (8/11/93):

- Communications Fraud Control Association: Last year loss due to unauthorized access to computer & telephones estimated to exceed $500M in U.S.; $2B Worldwide
- 58% customer want cross-platform security products
- 43% customers want Single Sign On Network Access Devices



Customer Requirements for Network Security

- **Security Functions:**
  - * 1. Single logon to multiple applications
  - * 2. Secure password exchange for multiple applications
  - \+ 3. Internet Firewalls
  - * 4. Authentication
  - * 5. Audit
  - * 6. Data Integrity
  - * 7. Data confidentiality
  - * 8. Authorization
  - 9. Access control
  - 10. Non-Repudiation
- **Customer Environment:**
  - * 1. Multi-platform: (OS/2, DOS, DOS/W, AIX/6000, SUN, HP) ...
  - * 2. Multi-protocol: (TCP/IP, NetBIOS, LU 6.2, IPX),...
- **Other Security Requirements**
  - \+ 1. Provide central security administration
  - \+ 2. Provide physical/PC security

*** = addressed by NetSP SLC**
+ = addressed by Other Products

Industry Authentication Technologies

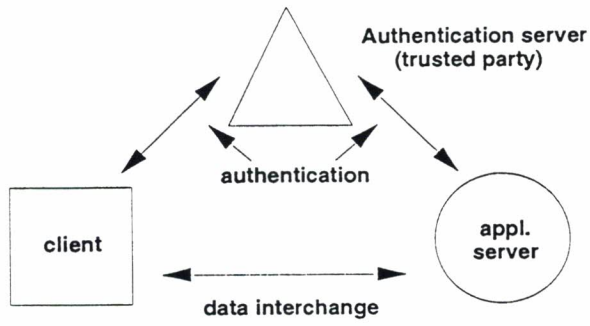
- **TWO-PARTY:**
  - End User and the Target System
  - Typically involves a Password

- **THIRD-PARTY:**
  - A separate, 3rd party carries out authentication
  - Uses a ticketing mechanism

  - Kerberos: Public domain software from MIT, developed in the 70's
  - KryptoKnight: Based on Kerberos, with many enhancements made
  - GSS-API: The Industry Standard

- **CARD BASED:**
  - Random Number Generator
  - Card Insert
  - "Something you have technology"

Third Party Authentication



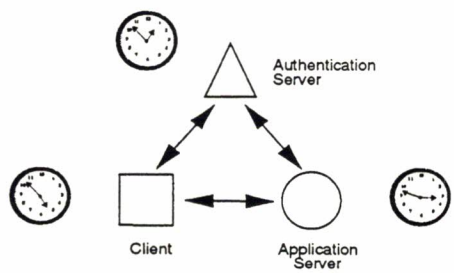
- **Clients/users and servers authenticated**
- **One (or more) security server required to be "trusted and secure"**

Network Security Program Key Features

- **Single Secure Signon:**
  - RACF MVS and VM Applications
  - OS/2 LAN Servers
  - Novell Netware Servers
  - Customer Applications
  - OS/2 Telnet Servers
- **Security for Distributed Applications**
  - Authentication
  - Data Integrity
  - Data Confidentiality
- **Audit**
- **Backup**
- **World Class Support**
- **Other Neat Features**

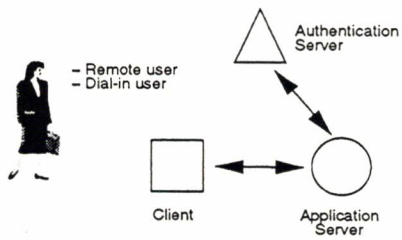
Superior Technology For Networks

1



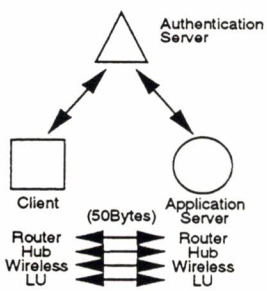
Does Not Require Clock Synchronization

2



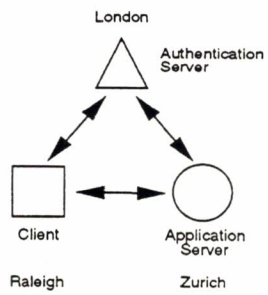
Client Does Not Require Connection to A.S.
(For Access to client/server applications)

3



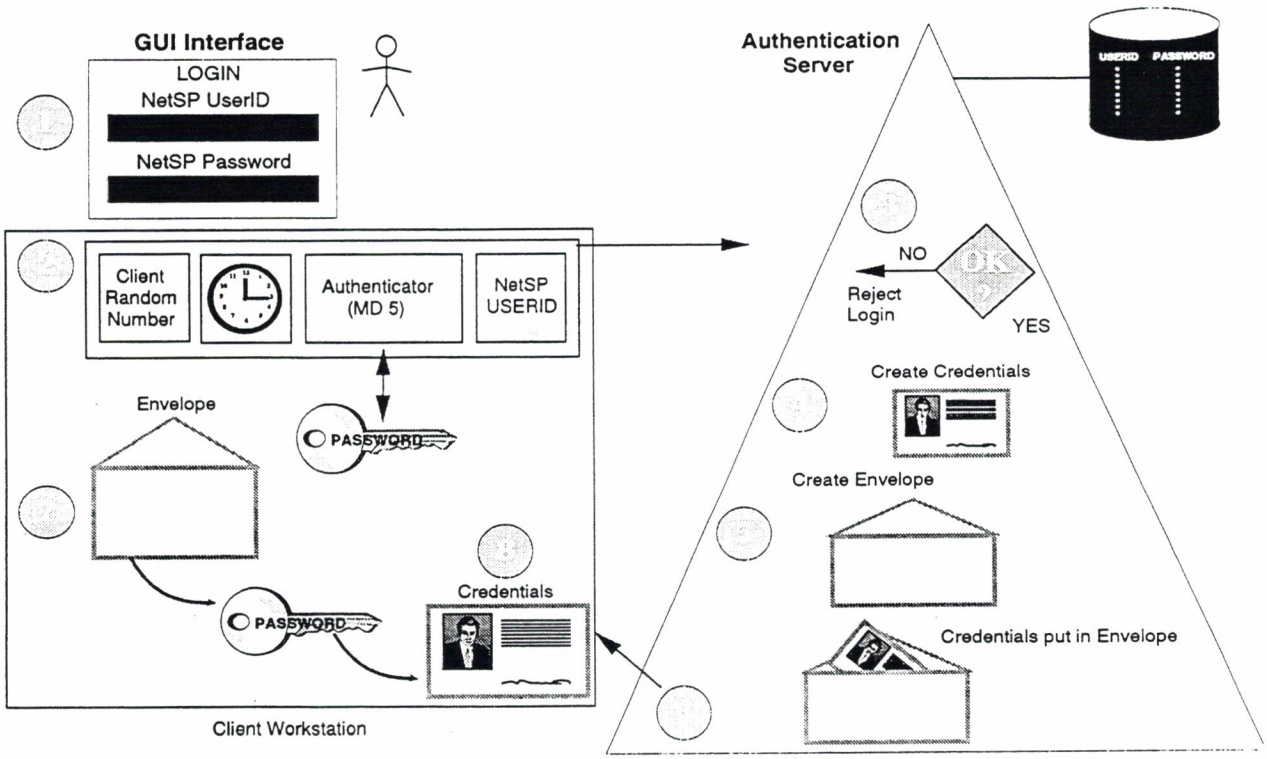
Ideal for Network-layer Authentication

4

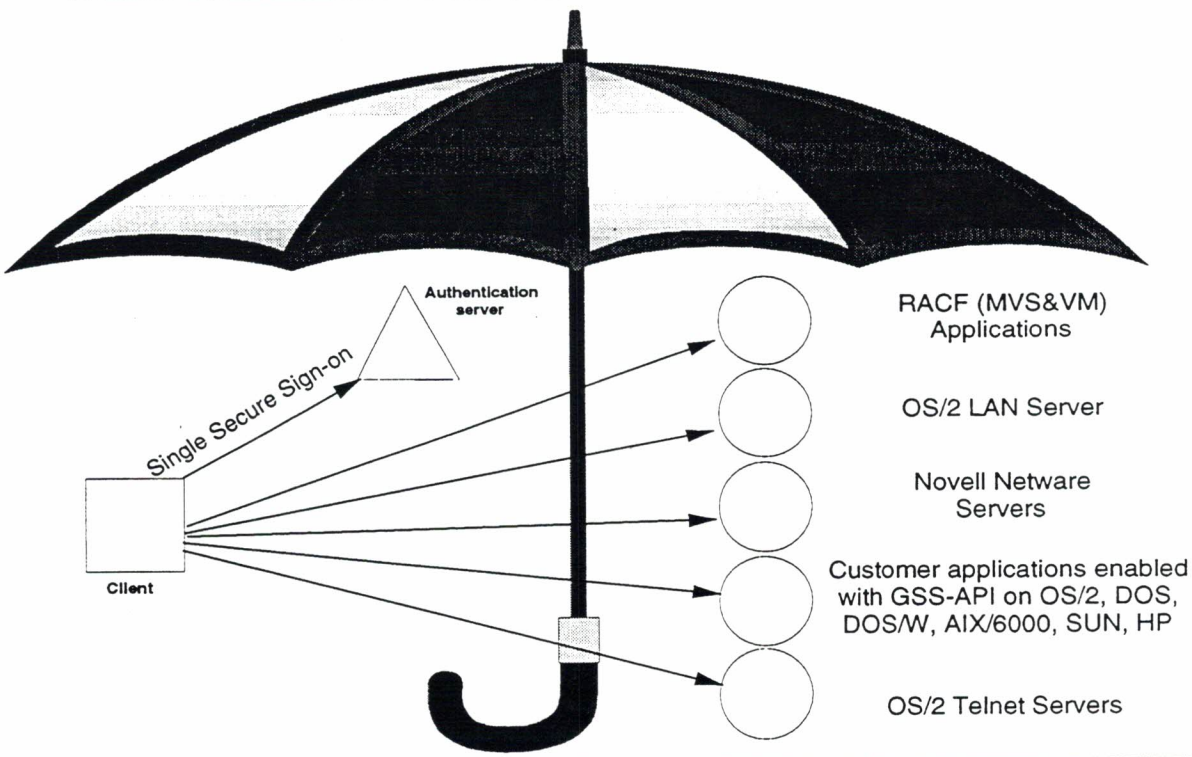


Freely Exportable

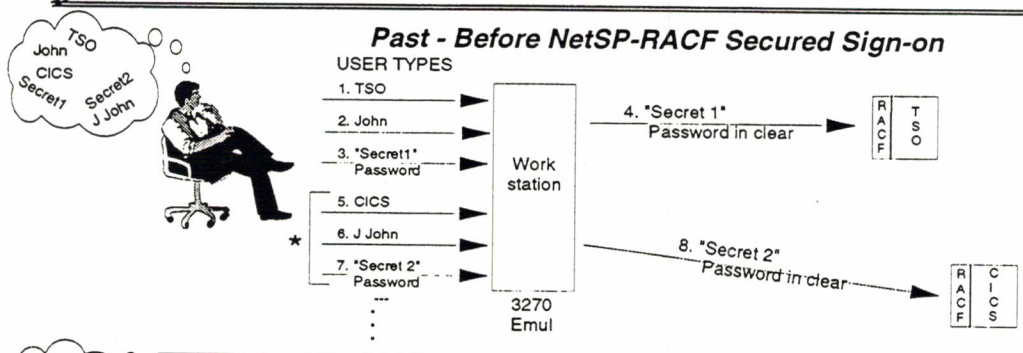
Single Secure Signon

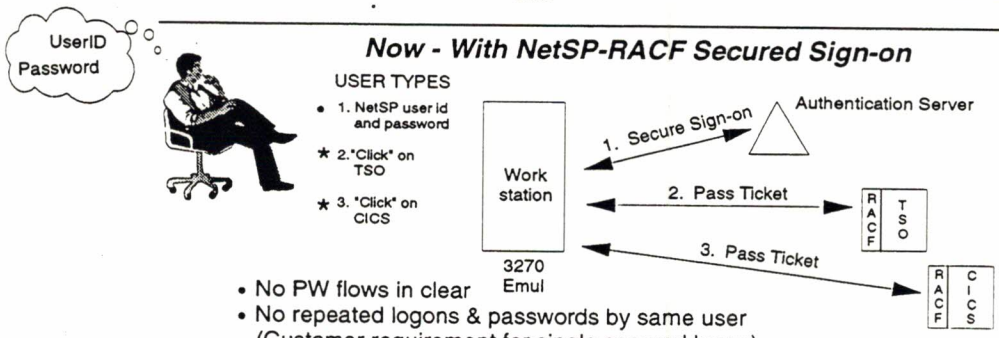


Single Secure Sign-on Umbrella



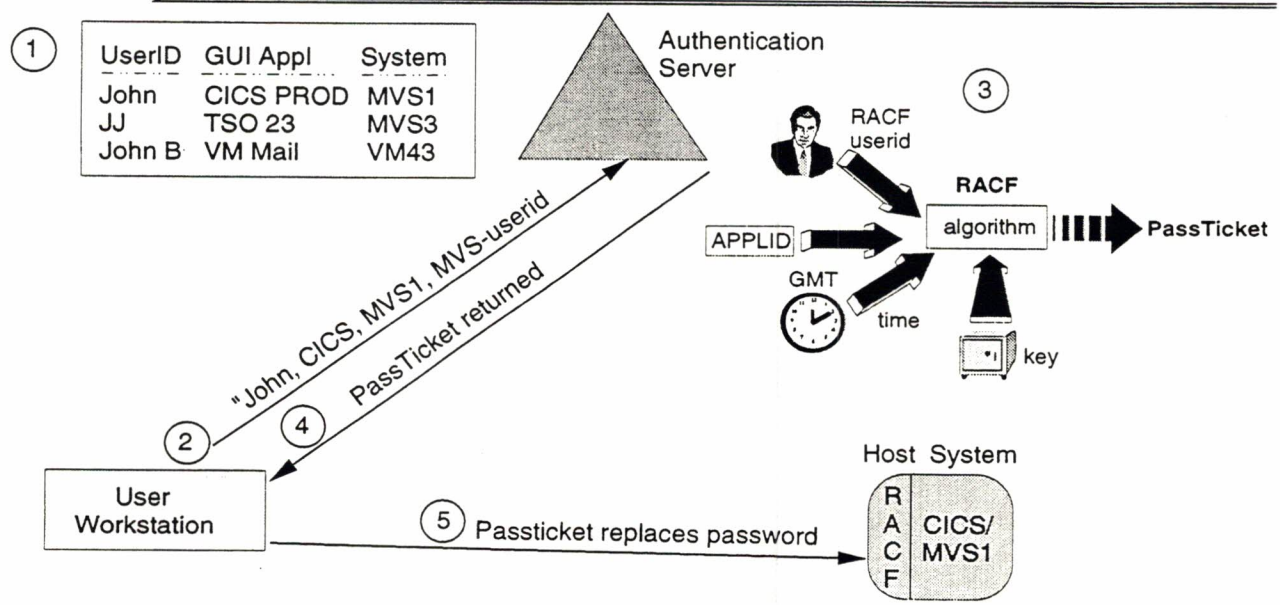
RACF Single Secure Sign-on Benefits: Usability & Productivity





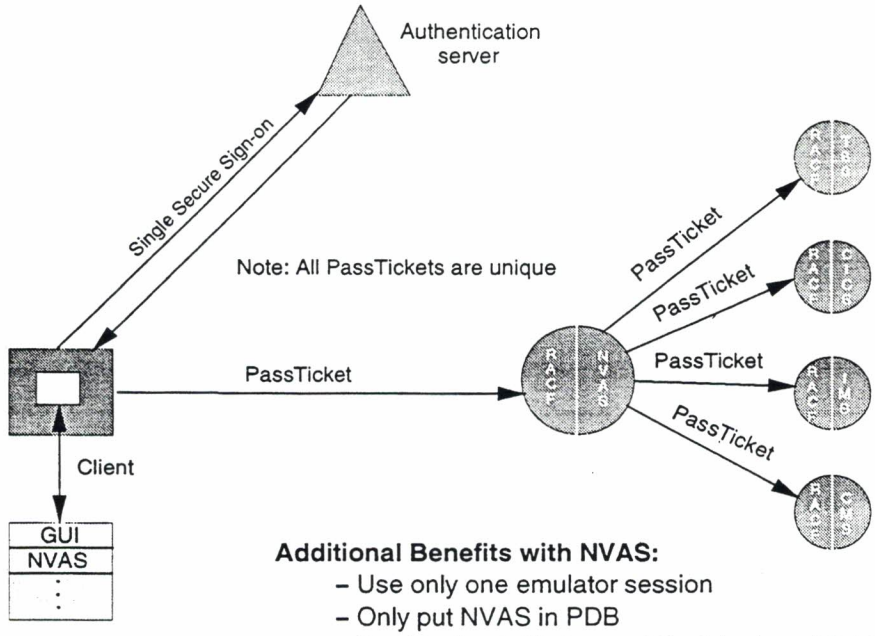
- No PW flows in clear
- No repeated logons & passwords by same user (Customer requirement for single secured logon)
- No changes to customer applications or 3270 emulators
- No need to remember how to navigate thru various screens

RACF Single Secure Sign-on Flows: Bullet - Proof Security



1. John chooses the RACF appl he needs to access
2. Client workstation requests a passticket
3. Authentication Server generates a PassTicket
4. A passticket is returned to the workstation
5. The passticket is given to RACF and John is logged on to CICS/MVS1

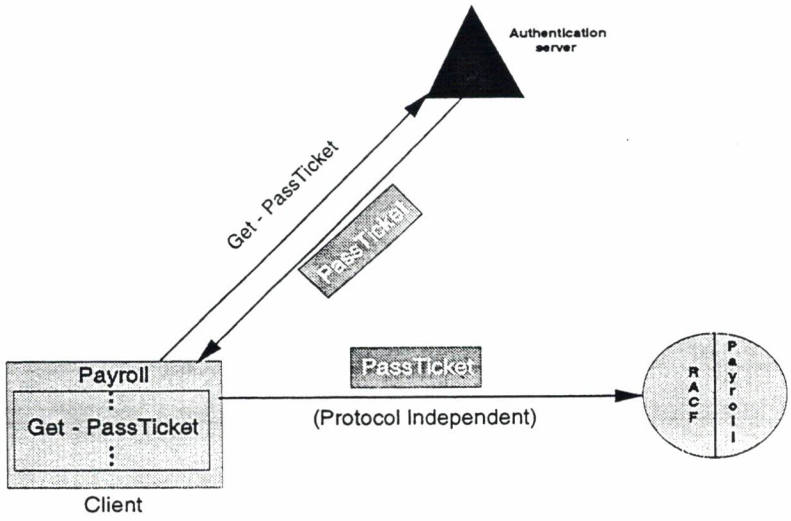
NetSP and NetView Access Service (NVAS) Complementary Products



Additional Benefits with NVAS:

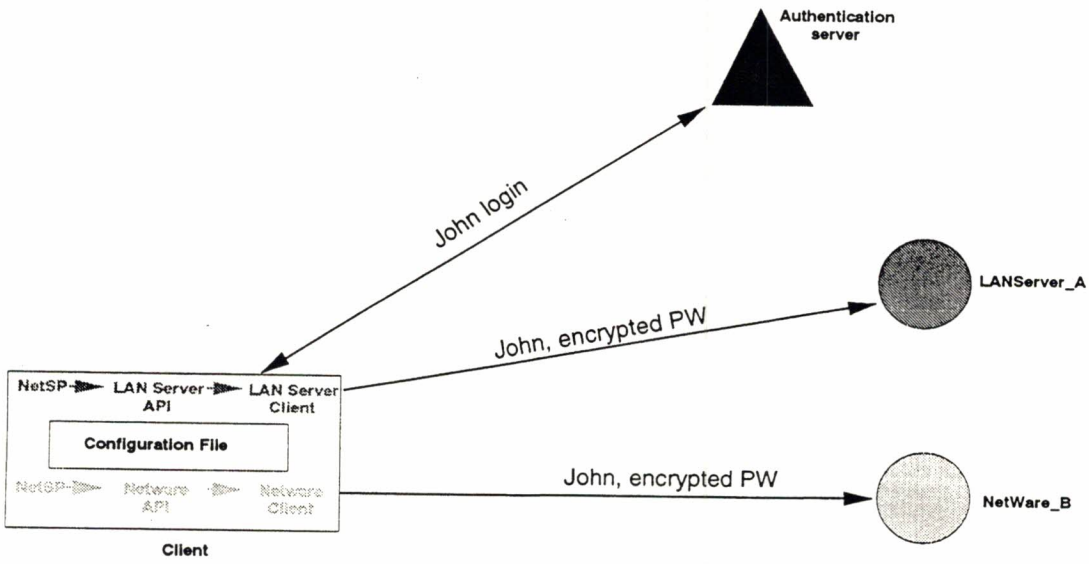
- Use only one emulator session
- Only put NVAS in PDB
- See Redbook "Enhanced Exploitation of RACF 1.9.2 Secured Single Signon", GG24-4184
- Requires modification to NVAS exits

PassTickets for non - 3270 Emulation



- Use a single API call in application
- Provides Single Signon for non-3270 applications

OS/2 LAN Server & Novell NetWare: Single Secure Signon

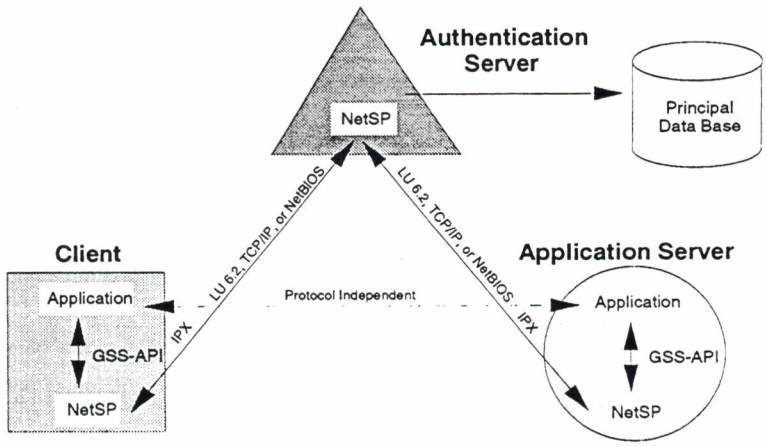


- Automatic login to servers
- Password Synchronized across servers securely

GSS API: Generic Security Service API

- **General consensus in the industry to support DEC's GSS API**
- **Kerberos V5.3 expected to have hooks in for GSS API**
- **Internet: Adopted 2Q93 as an RFC (RFC 1508, 1509)**
- **OSF: DCE next release item**

GSS-API Flows

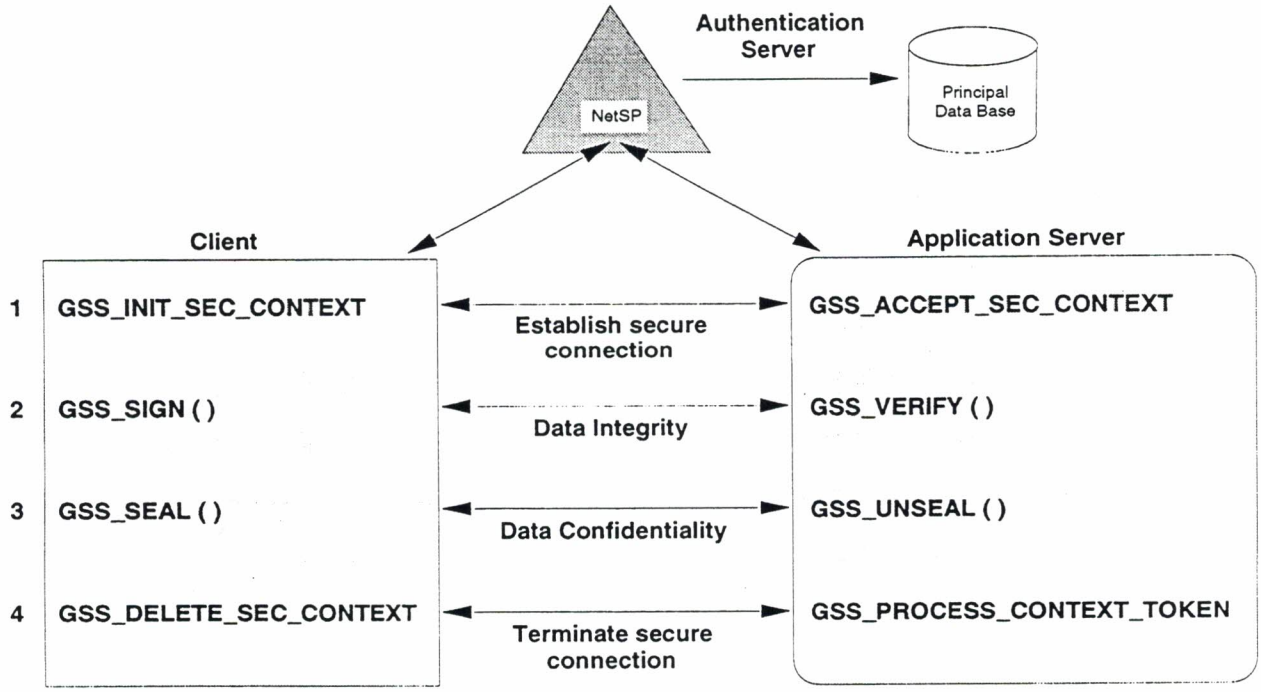


1. Application requests security services from NetSP using GSS-API

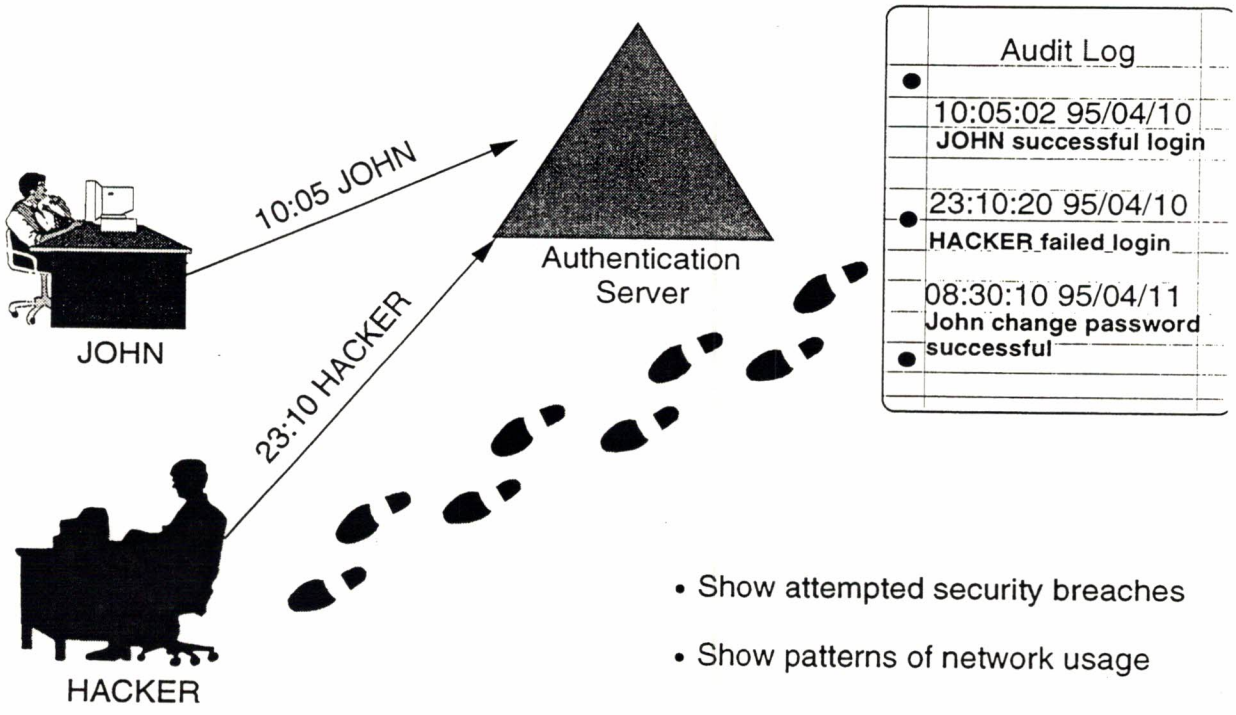
2. NetSP performs the request and returns the result to the application program

3. Application communicates directly with its peer application (NetSP not involved)

GSS-API Example

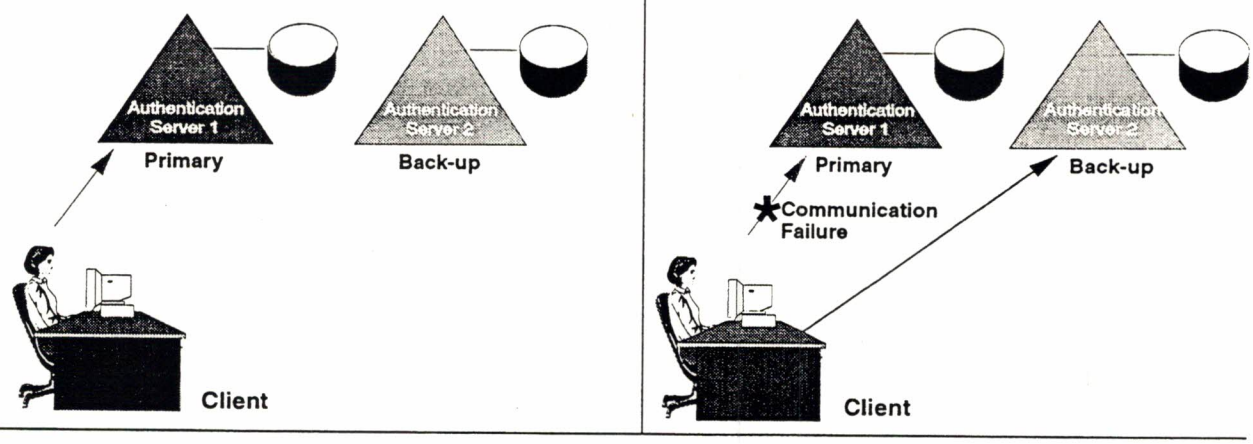


Audit



- Show attempted security breaches
- Show patterns of network usage

Back-up Capability

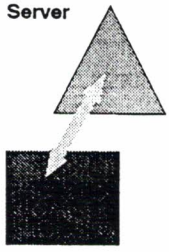


Re-Routing Authentication Server Traffic Transparent to end user
Recommendations:

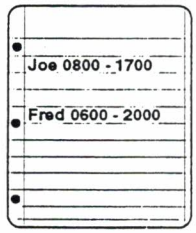
- Authentication Server dedicated to NetSP
- Provide back-up power supply
- Lock-up Authentication Server

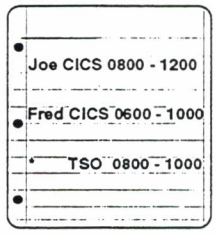
Other Neat Capabilities

1. Authentication Server



Client

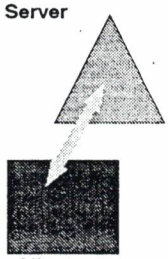




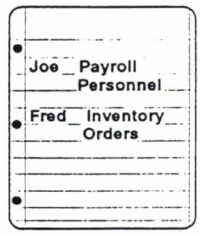
Login Exit:
RACF Signon Exit:

(Limit conditions for Login & RACF Signon)

2. Authentication Server

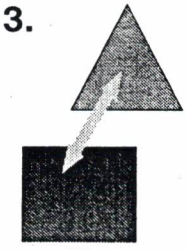


Client



Authentication Exit
(Authorization)

3.



Client

Password Change Exit
(Enforce password quality)

Audit Exit
(Notify security administrator)

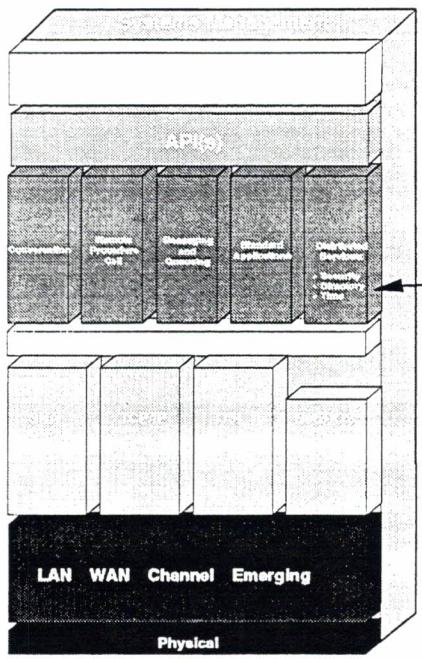
Networking Blueprint for the 90's & Beyond

IBM Networking is

- Multiprotocol
- Open and standard
- Peer-to-peer
- Addressing evolving subnetworking technology

Key Objectives

- Extend current customer networks
- 'Future-proof' customer investments
- Integration of wide area networks & multiprotocol local area networks
- Cost effective standards-based high speed transmission
- Distributed computing infrastructure



Support

- **It is provided by the experts!**
  - One-day class taught by our developers
  - Service by calling 1-800-423-5511 and speaking to our developers
  - The NetSP Forum on TalkLink on IBMLink

NetSP Business Case

One customer estimates:

1	hour per year resetting passwords (User plus help desk)
21	hours per year signing on and maintaining passwords (Based on 5 min/day)
22	hours/year/user
x $10/hr	
$220	per user per year

Saving Time and Money

"The average user of four applications spends about 44.4 hours per year just logging on to those applications. That's a cost of about $1,082 for each user.

Communications Week, 12/5/94

Pricing and Packaging

Authentication Server * 60 - Day trial period available

- $2495 for first copy
- $2245 for additional copies

Client

- $120 for first copy
- $99 for additional copies

For Customers who participate in the VALU* Program

Number of Clients	% Discount	Price per Client
1	0	$120
2 - 249	0	$99
250 - 499	30	$69
500 - 999	34	$65
1000 - 2999	38	$61
3000 - 4999	43	$56
5000 +	50	$49

* Allows you to aggregate different products to obtain the maximum discount.

NetSP Direction

1. Extend Single Secure Signon
   - Unix native security
   - AS/400
   - ISV's
2. LAN Distance Integration
3. Primary/Backup Synchronization
4. Security Dynamics Integration
5. Secure remote administration
6. Enhanced Audit
7. Different levels of administrative authority

Interoperability with Other Related Products

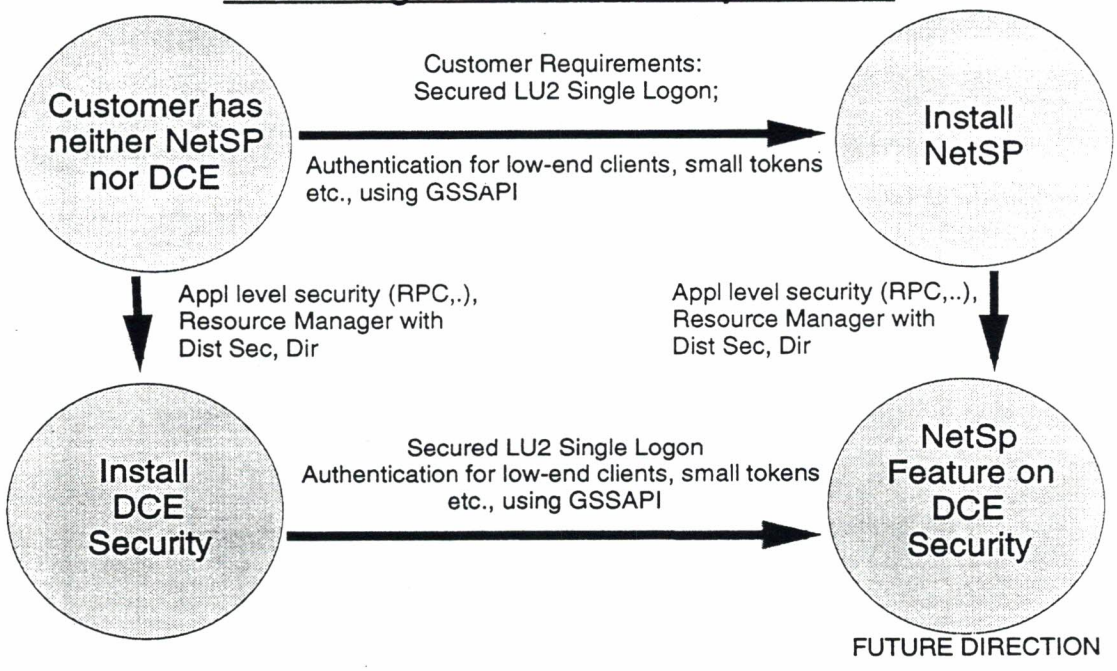
Company	Product	Contact	Target Availability
Argo	Bank Pro	Bala Shagrithaya 214-386-4949	
Computer Associates	ACF2 6.1 + OpenMVS APAR TOPSecret Beta 5.0 CA-TeleView Beta CA-VMAN Beta	Support Number 800-645-3042	
Fischer International	Watchdog	Ranae Rousse 813-643-4803	
IBM	Distributed Security Manager/MVS	Chuck Mahood 914-435-6965	
Legent	TPX	Dave Connel 412-494-1784	
Maxim Systems	Maxim	Gary Greathouse 703-761-0416	
Mergent International	PC/Dacs	Tara Hills 203-257-4223	
PC Dynamics	Menu Works Total Security	Bruce Clay 818-889-1741	
Security Dynamics	ACE Server	Scott Baumann 919-254-7416	
Sterling Software	VM Secure	Steve Drill 703-264-8303	
Candle Corp	CL/SUPERSESSION	Nigel Williams 708-954-3888	
Quality Systems Development B.V.	Q-SMS	Mark Kristel 011-31-34655144	

Network Security Program and DCE

- **NetSP is IBM's value add to the DCE products**
  - Provides a Single Secure signon to RACF Applications
  - Provides the capability to do network layer authentication
  - Provides the capability to be authenticated when the user has no connection to the Authentication Server
  - Provides callable security services via the GSS-API

NetSP and DCE Security

Positioning with Customer Requirements



Reference Documents

- **SC31-6500 Network Security Program Developer's Guide**

- **GG24-4149 Using Network Security Program (Redbook)**

- **SC23-3765 RACF Passticket, Version 1 Release 9.2**
  + APAR OY65281 or APAR VM56690

  Note: above function matches NetSP V1R1 passticket support. Additional RACF Passticket APAR-OW03024 requires NetSP APAR IR25746 for V1R1

- **GG24-4184 Enhanced Exploitation of RACF 1.9.2**
  Secured Single Signon (Redbook)

this page intentionally left blank

CS 0901
OS/2 Database Design and Programming

Rob Veitch
Watcom
Waterloo, Ontario

Introduction to Database Design and Programming

OS/2 World
July 20, 1995

Rob Veitch, WATCOM

Introduction

- What is DB2/2?
  - Its role in DRDA
- Relational Database Design
  - The theory
- Relational Database Programming
  - SQL
- Order Entry Example
  - A simple example

What is DB2/2?

Relational Database Management System (RDBMS)
- Server in client/server
- MIS and OLTP
- Engine for distributed applications that involve shared data and multiple users

DB2/2 features

- 32 bit SQL engine, referential integrity, remote procedure calls
- Single user & client/server editions
- APIs for C, Fortran, Cobol, and REXX
- Administration tools
- DRDA access

What is DRDA?

- Distributed Relational Database Architecture
  - DB2/2, DB2/6000, SQL/400 DB2 for AS 400, SQL/DS DB2 for S/370 VM, DB2 for S/370 MVS
- Distributed Database Connection Services (DDCS/2) is the gateway to DRDA for DB2/2
- Portability/scalability of applications

Relational Database Design

- Background
  - Relational model developed at IBM's San Jose Research Lab in 70s by E.F. Codd
  - Replaced hierarchical (pointers) with relational (reference by value)
  - Hierarchical hard to maintain
  - Relational slow

Tables

- Data organized into tables and columns within tables
- Order of rows not important, rows identified by key values
- Tables related through matching key values
- Normal forms avoids duplicate data

Columns

- Column order is not significant, identified by unique name
- Number of columns is fixed, all entries in a column are of the same type and represent the same kind of thing.

Rows

- Row order is not significant, identified by the primary key (unique column values)
- Can be multiple columns in a primary key.
- Number of rows is not fixed
- No duplicate rows

Relations

- Tables are related by matching columns
- Common columns form the primary key of one table
- Called a foreign key in the other table
- Key joins

ER diagrams

- Diagram of entity (table) relations.
- Shows the cardinality ratio of the relation:
  - One to one --------
  - One to many -------->
  - Many to many <-------->
- One side is primary key, many side is foreign key
- Must split many to many by adding another table

Order Entry Tables

- ER Diagram
- 3rd Normal Form
  - Value of all non-key columns depends on the primary key and nothing but the primary key

Relational Database Programming

- SQL
  - Structured Query Language
  - Also developed by IBM at San Jose in 70s
  - ANSI standards:
    SQL89 (most engines)
    SQL92 (standard exists)
    SQL3 (standard in progress, e.g. stored procedures)

Interactive/Embedded

- SQL verb and object
- Data definition: CREATE, DROP
- Data manipulation: SELECT, UPDATE
- Data control: COMMIT, ROLLBACK, GRANT
- Embedded in procedural languages: PREPARE, EXECUTE

Dynamic/Static/Stored procedures

- Dynamic is "interpreted"
- Static is precompiled
  - Plan of execution can be made in advance and bound to server
- Remote procedures
  - Online transaction processing
  - Non-standard

Views

- SELECT
  - Select columns, tables, matching values, ordering, column functions
  - Cursor (forward only in DB2/2)
- Joins
- Views
- Indexes

Updates

- Update, Insert, Delete
  - Updates on single tables only (DB2/2)
  - Matching value(s) or current cursor
- Unit of work
  - COMMIT or ROLLBACK
- Referential integrity
  - Delete rules: Restrict, Cascade, Set NULL

Transactions

- Multiple users
- Isolation levels and currency
  - Read repeatable: every read row is locked
  - Cursor stable: current row is locked
  - Uncommitted read
- Deadlock detection
- GUI considerations

Good books

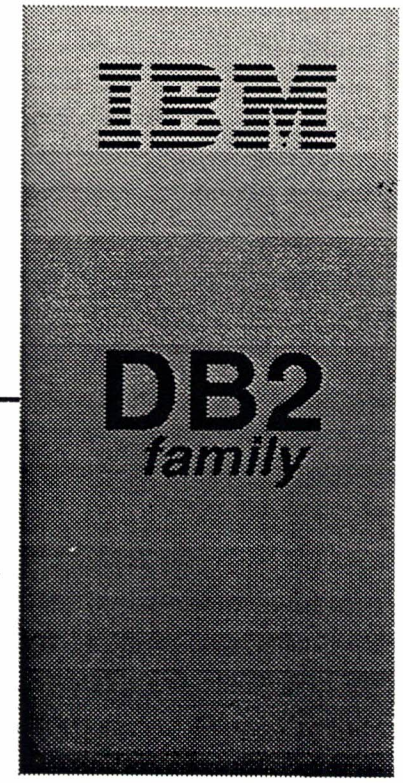
- Client/Server Programming with OS/2 2.1
  - Robert Orfali/Dan Harkey
  - Van Nostrand Reinhold, New York
  - ISBN 0-442-01833-9
  - 3rd edition, 1993
- Client/Server Survival Guide with OS/2
  - 1st edition, 1994

Order Entry Application

- **Built with VX-REXX Client/Server Edition**
- **VX for DB/2!**

CS 0902
DB2 Family: DB2 Technology at Work

Dan Britton
IBM



DB2 Family Update

DB2 Technology at Work

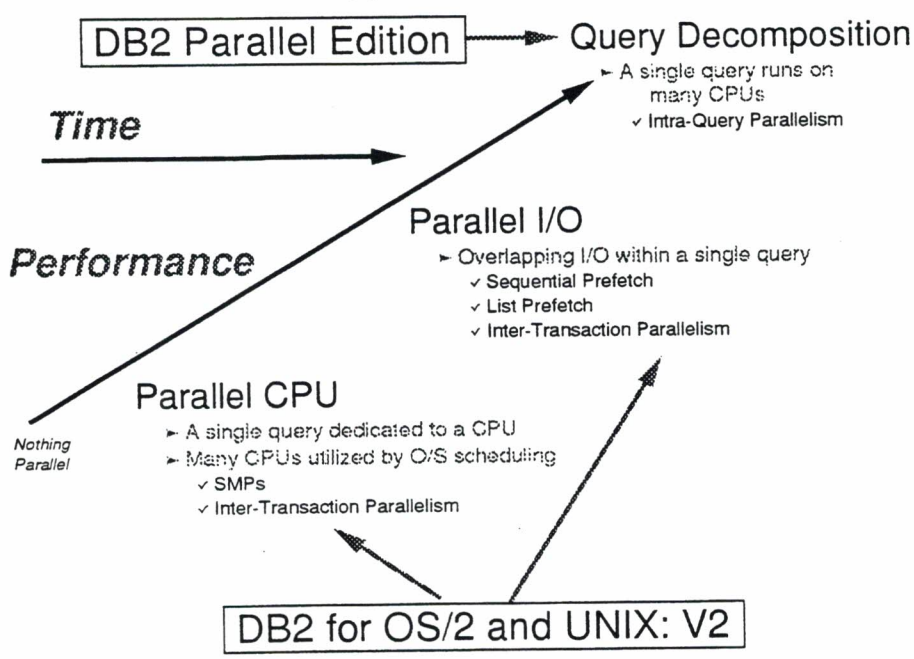
Dan S. Britton
DB2 Family Specialist
IBM Santa Teresa Laboratory
BRITTON@VNET.IBM.COM
512-823-8937

DB2 for Workstations

- Current Products
  - DB2 for HP-UX, Version 1
  - DB2 for Sun Solaris, Version 1
- Planned Products
  - DB2 for HP-UX, Version 2
  - DB2 for Sun Solaris, Version 2
  - DB2 for Siemens SINIX, Version 2
  - DB2 for NT, Version 2
  - Visual DB2
- Under Consideration
  - DB2 for DEC
  - DB2 for NCR

IBM® Software
© Copyright IBM 1995

Parallel Technology Progression



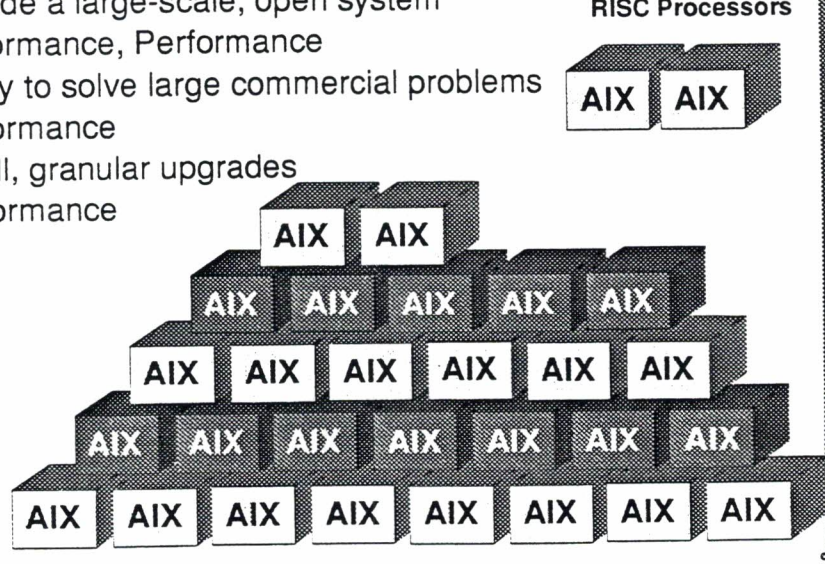
IBM DB2 Parallel Edition

© Copyright IBM 1995

IBM® Software

DB2 Parallel Edition: Why ?

- ✓ Performance, Performance, Performance
- ✓ Provide a large-scale, open system
- ✓ Performance, Performance
- ✓ Ability to solve large commercial problems
- ✓ Performance
- ✓ Small, granular upgrades
- ✓ Performance

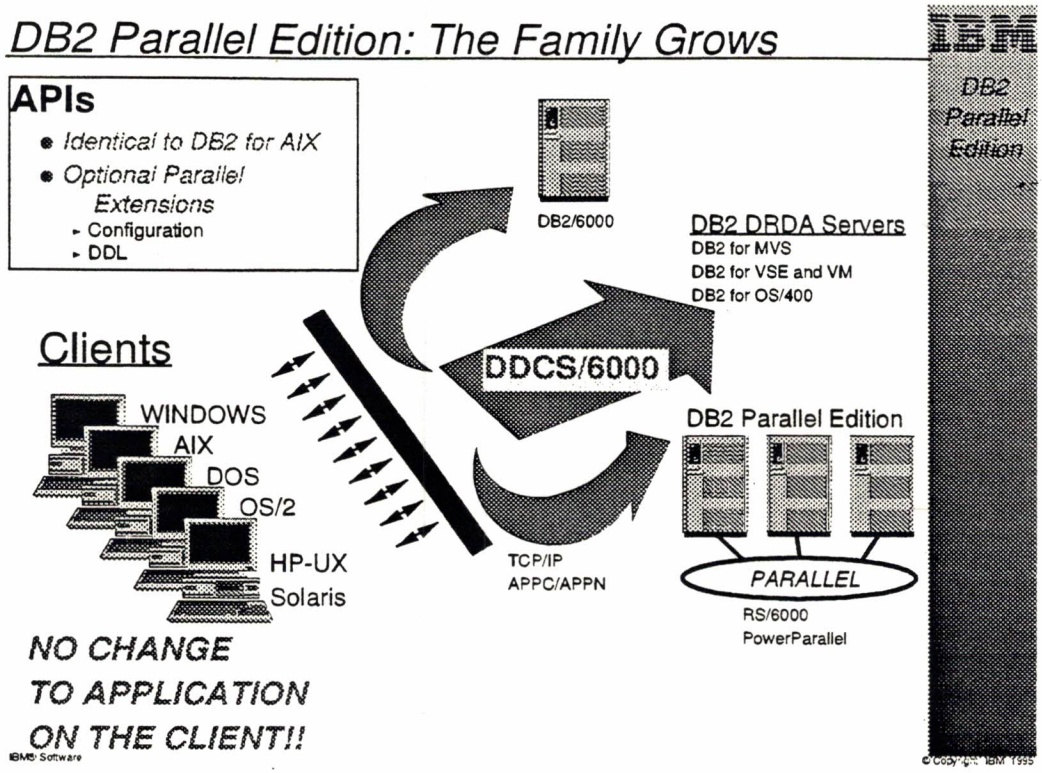


IBM DB2 Parallel Edition

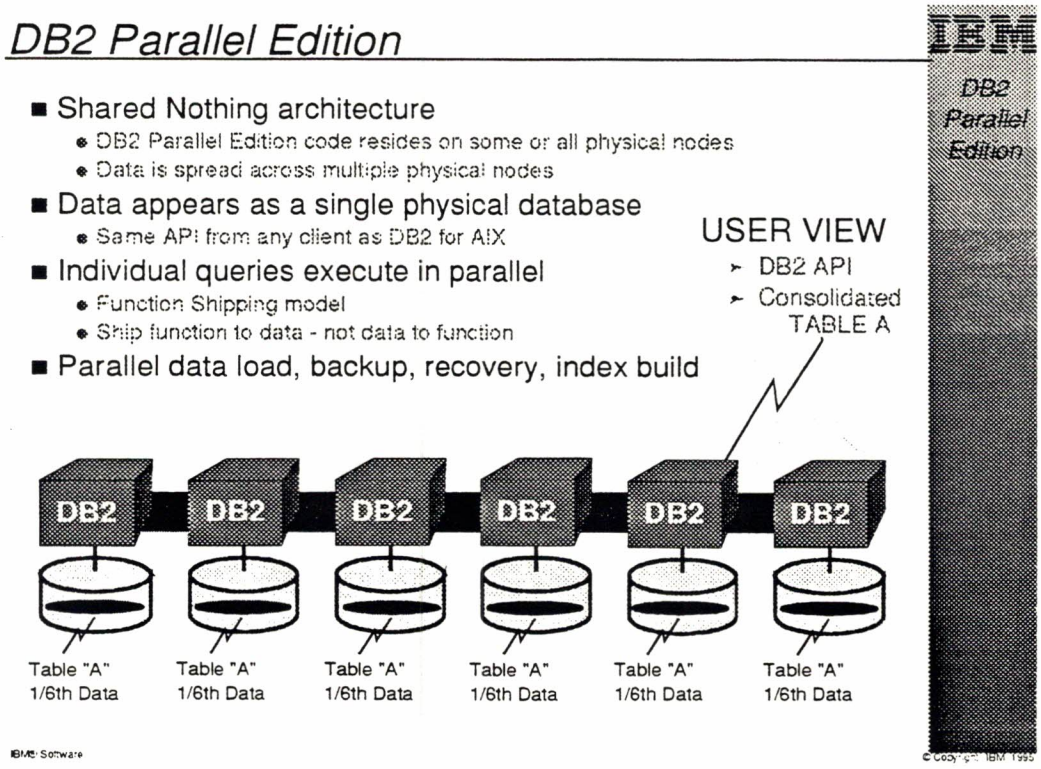
© Copyright IBM 1995

IBM® Software

DB2 Parallel Edition: The Family Grows

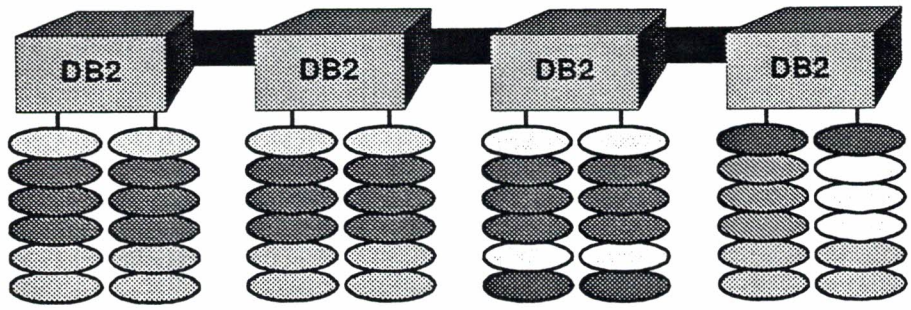


DB2 Parallel Edition



DB2 Parallel Edition - Data Distribution

- Nodes are grouped into 'Nodegroups'
  - increased granularity of data placement
  - Allows a table to span one or more physical nodes
  - CREATE NODEGROUP ***PURPLE*** ON NODES *1* TO *3*
  - CREATE NODEGROUP ***RED*** ON NODES *3* TO *4*
- Tables exist in Nodegroups
  - CREATE TABLE ***CUSTOMER*** (*C1* INTEGER, *C2* CHAR(*20*)) IN ***RED*** PARTITIONING KEY (*C1*) USING HASHING
  - Tables can be co-located in a Nodegroup
- Rows are hash distributed within their Nodegroups
  - Partitioning key defined by DBA



IBM® Software

IBM DB2 Parallel Edition

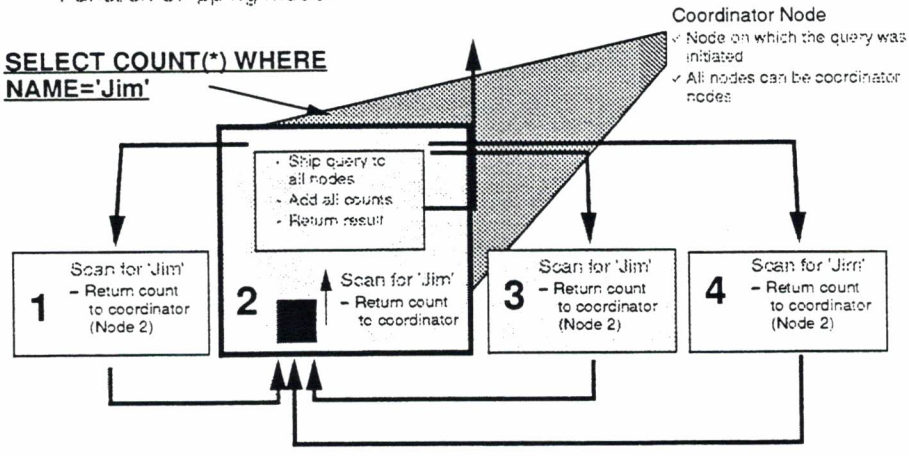
© Copyright IBM 1995

DB2 Parallel Edition: Execution Model

Intra-Query Parallelism

Individual query is run in parallel

Function Shipping model



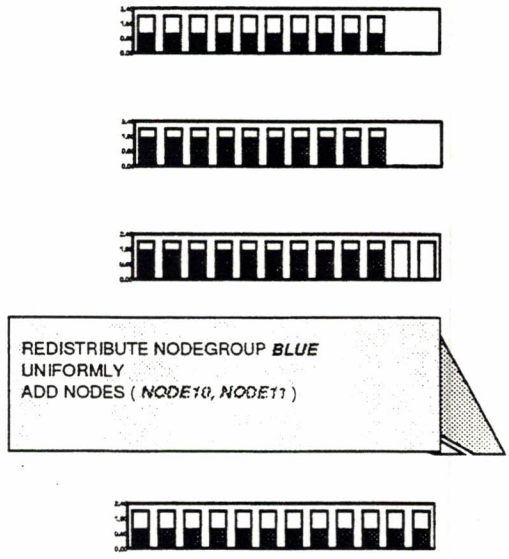
Similar model for UPDATEs, DELETEs & INSERTs

IBM® Software

IBM DB2 Parallel Edition

© Copyright IBM 1995

DB2 PE Rebalance



Nodegroup BLUE is in proper balance
Sufficient processing power available
Everbody is happy!!!

Time goes by and Nodegroup grows
People are not so happy
Time to add more processing power

Add nodes to the configuration
ADD NODE *NODE10*
ADD NODE *NODE11*
Power added, but not exploited
People are not so happy

REDISTRIBUTES ROWS
TABLE BY TABLE

Nodegroup BLUE is in proper balance
Sufficient processing power available
Everybody is happy!!!

IBM® Software

IBM DB2 Parallel Edition

© Copyright IBM 1995

DB2 Parallel Edition - Parallel Algorithms

- Parallel Queries
- Parallel Transactions
- Parallel Utilities
  - Index creation
  - Redistribute
  - Runstats
  - Reorg
  - Load
- Parallel Scans
  - Relation scan
  - Index scan
- Join Algorithms
  - Collocated
    - Tables are co-located
    - Partitioning key = join predicate
    - Join done locally
  - Redirected
    - One tables' partitioning key = join predicate
    - Apply predicates
    - Direct answer set to appropriate node
  - Redistributed
    - Partitioning key not equal to join predicate
    - Apply predicates
    - Redistribute answer set
  - Broadcast
    - Same conditions as redistributed
    - Large ratio of table size
    - Broadcast smaller table to nodes containing larger table

IBM® Software

IBM DB2 Parallel Edition

© Copyright IBM 1995

Broad Range of Queries Tested

IBM DB2 Parallel Edition

- ✓ 2 Way Join (On-Line Services)
  - ► 15,750,141 rows x 3,309,38 rows
  - ► 8 SP1 nodes: 5 minutes, 38 seconds
- ✓ 3 Level Correlated Subquery (Consumer Products)
  - ► 60,000,000 rows x 280,000,000 rows x 90,000,000 rows
  - ► 64 SP1 nodes: 11 minutes, 43 seconds
- ✓ 3 Way Join (Shipping)
  - ► 195,154,800 rows x 15,968,200 rows x 15,207,000 rows
  - ► 8 SP2 nodes: 12 minutes, 35 seconds
- ✓ 6 Way Join, 3 Joins Self Referencing, Using EXCEPT
  - ► 200,000,000 rows x 80,000,000 rows x 100,000 rows x 200,000,000 rows x 80,000,000 rows x 100,000 rows
  - ► 46 SP1/SP2 Nodes: 2 minutes, 12 seconds

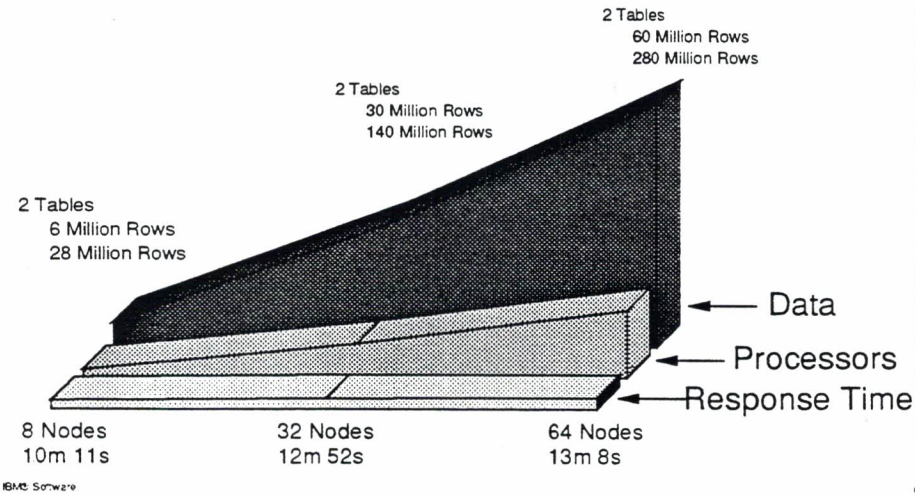
IBM® Software

© Copyright IBM 1995

Demonstrated Excellent Scale Up

IBM DB2 Parallel Edition

More Data + More Processors = Consistent Response Time



IBM® Software

© Copyright IBM 1995

DB2PE: More Than Just Extended DB2

IBM DB2 Parallel Edition

- DDL syntax & semantics for parallel
- DML semantics & execution for parallel
  - All SELECTs, INSERTs, UPDATEs, DELETEs
  - Package/plan structures
- Optimization for parallel
  - Based on research & design at Almaden & Hawthorne
  - CPU, I/O and network costs considered
  - Locking done locally, deadlock detection globally
- Parallel utilities
- High speed inter-node communication
- Full transactional semantics across nodes
- Parallel run-time infrastructure and process model
- Single system image and programming interface

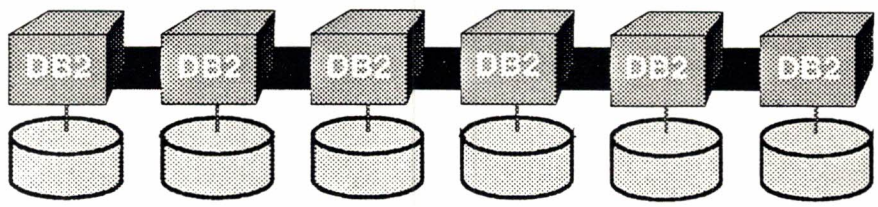
IBM® Software

© Copyright IBM 1995

Key Points

IBM DB2 Parallel Edition

- ✓ What Parallelizes ?
  - Everything !
- ✓ Size/Capacity Support
  - 100GB tested; 300GB test underway
- ✓ Query Performance
  - Outstanding due to parallel optimization of SQL
  - Wide range of real customer, real business problem SQL tested



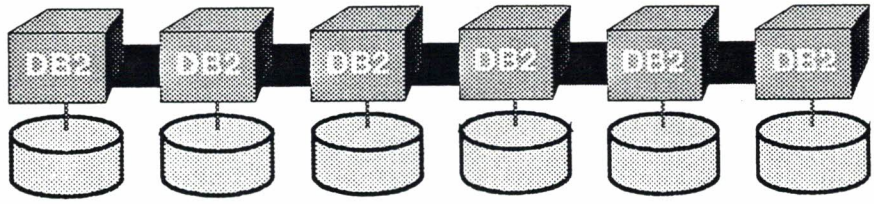
IBM® Software

© Copyright IBM 1995

Key Points

IBM DB2 Parallel Edition

- ✓ Scaleability
  - Demonstrated data growth from 10 GB to 300 GB
  - Great speed up results
  - Excellent scale up results
- ✓ Concurrency
  - 30 concurrent DSS users already tested
  - Architected to function ship
- ✓ Loading Data
  - Proven 2 GB per node per hour



IBM® Software
© Copyright IBM 1995

DB2 for MVS Version 4 Highlights

IBM DB2 for MVS

- ■ Client/Server
  - Stored Procedures
  - Large Network Support
- ■ Availability
  - Locking: Index, Row, Read-Through
  - Full Partition Independence
  - Data Sharing
  - Concurrent Copy Managed by DB2
- ■ Performance
  - Data Sharing
  - CPU Parallelism
- ■ Usability
  - Outer Join
  - Table Check Clause, Default Values
  - Cancel Thread, Catalog Reorg

IBM® Software
© Copyright IBM 1995

DB2 for AS/400 Highlights

- ANS/ISO 92 Level 2 Conformance, IBM SQL
  - Stored Procedures
  - Long Column Names (30 Characters)
  - Declarative Referential Integrity
- Triggers
- System Wide Catalog
- Multimedia Support
- Protocol Independence
- C2 Security

IBM® Software
© Copyright IBM 1995

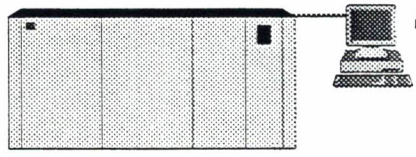
DB2 for AS/400 Highlights

- Enhanced Performance
  - Remote SQL - 300%
  - Query Governor
  - SQL Optimizer
  - Parallel Data Access
  - Performance Analyzer
  - Blocked Fetch & Insert
  - Multiple Program Selectable Isolation Levels
- DRDA Level 2
  - Two Phase Commit
  - Distributed Unit of Work
- Open Access (ODBC)

IBM® Software
© Copyright IBM 1995

Islands of Data

- Multiple technologies
- Poor data accessability
- Unreconciled data
- Uncontrolled redundancy
- Extract processing
  - Inconsistency
  - Limited enhancements
  - Unmanaged
- Many gateways & tools



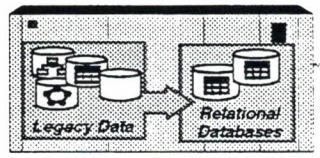
IBM Data Replicatio

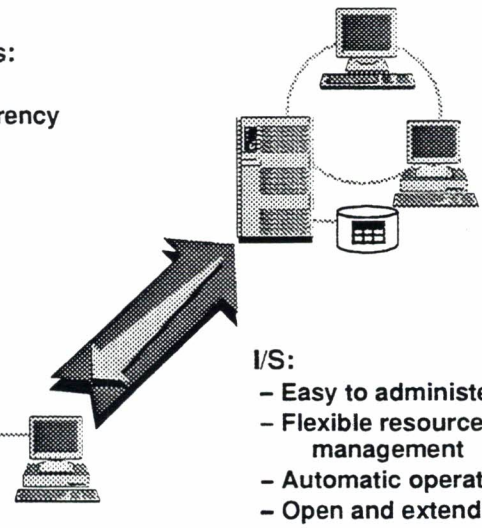
IBM® Software

© Copyright IBM 1995

Enterprise Needs

End-Users or Applications:
- **Access to all data**
- **Known and controlled currency**
  - ▸ **Point in time**
  - ▸ **History**
- **Customized formats**
  - ▸ **Detailed**
  - ▸ **Derived**
  - ▸ **Summarized**
- **Consistent environment**





I/S:
- **Easy to administer**
- **Flexible resource management**
- **Automatic operation**
- **Open and extendible**

IBM Data Replicatio

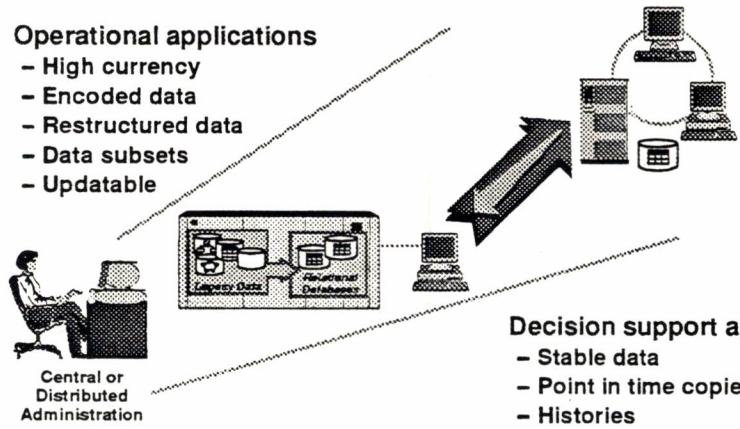
IBM® Software

© Copyright IBM 1995

The IBM Data Replication Solution

IBM Data Replicatio

Operational applications
- High currency
- Encoded data
- Restructured data
- Data subsets
- Updatable



Central or Distributed Administration

Decision support applications
- Stable data
- Point in time copies
- Histories
- Data derivation
- Data aggregation
- Data transformation
- Read only

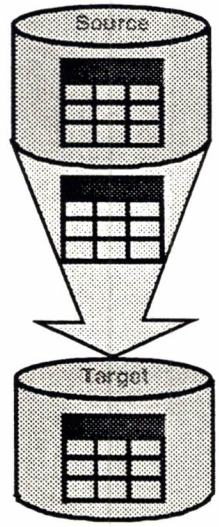
IBM® Software

©Copyright IBM 1995

Refresh

IBM Data Replicatio

Replace target data with a new copy from the source



- Simplicity
- Small data volumes
- Bulk load
- Infrequent copies

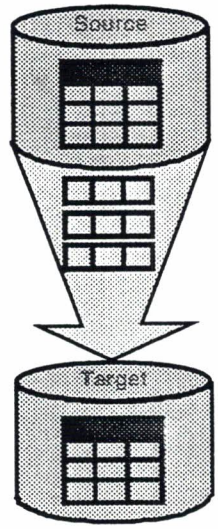
IBM® Software

©Copyright IBM 1995

Update

IBM Data Replicatio

Apply changes in the source to the target database



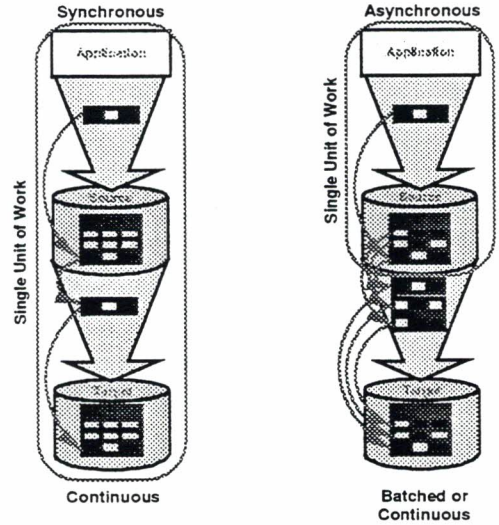
- Large databases
- Support for history
- Network load balancing
- Frequent copies

IBM® Software

©Copyright IBM 1995

Update

IBM Data Replicatio



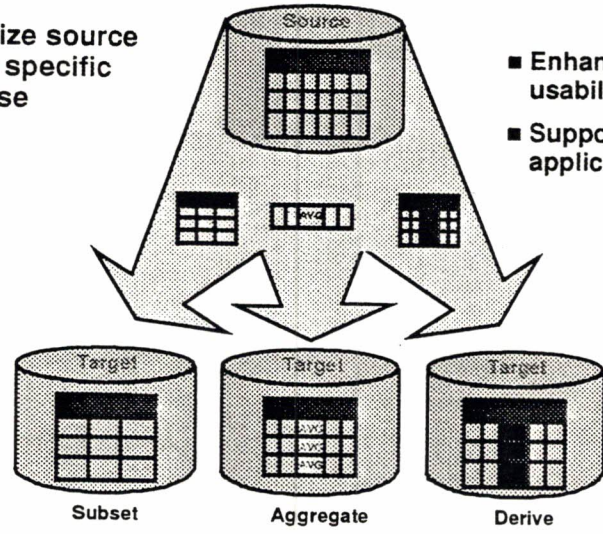
IBM® Software

©Copyright IBM 1995

Enhancement

IBM Data Replicatio

Customize source data for specific target use



- Enhances data usability
- Supports unique application needs

IBM® Software
©Copyright IBM 1995

Administering Data Replication

IBM Data Replicatio

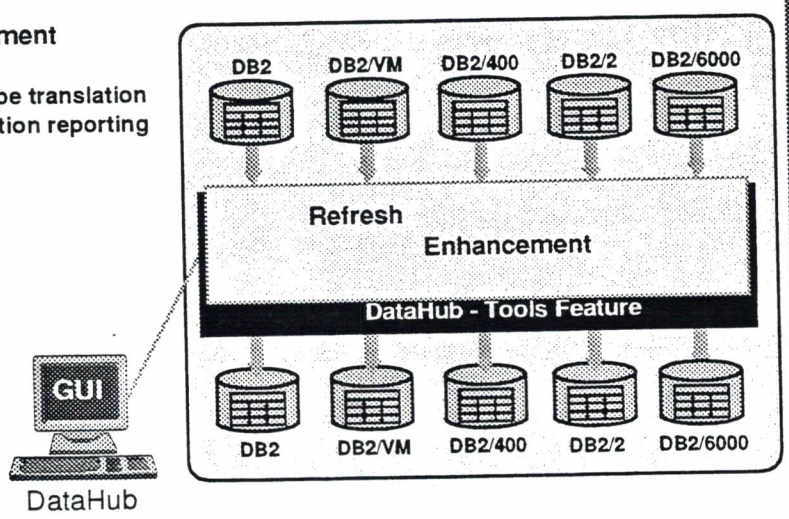
Built on DataHub
- Single control point for data administration
- GUI for source and target definitions
- Automated table creation and loading

Advisor Tools	Replication Tools	Vendor Tools	Self-Written Tools
Integration Platform for Tools			

IBM® Software
©Copyright IBM 1995

Datahub - Tools Feature

- Enhancement
  - Subset
  - Data type translation
  - Implication reporting

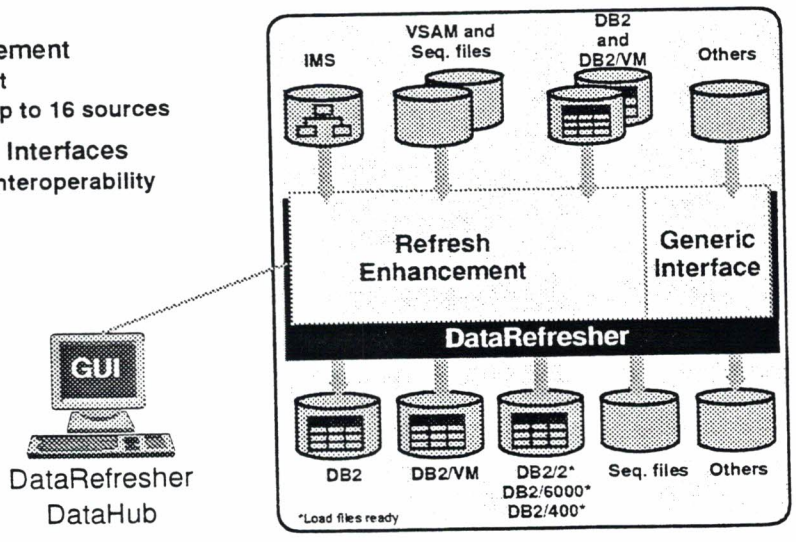


IBM Data Replicatio

IBM Software
© Copyright IBM 1995

DataRefresher

- Enhancement
  - Subset
  - Join up to 16 sources
- Generic Interfaces
  - Tool interoperability

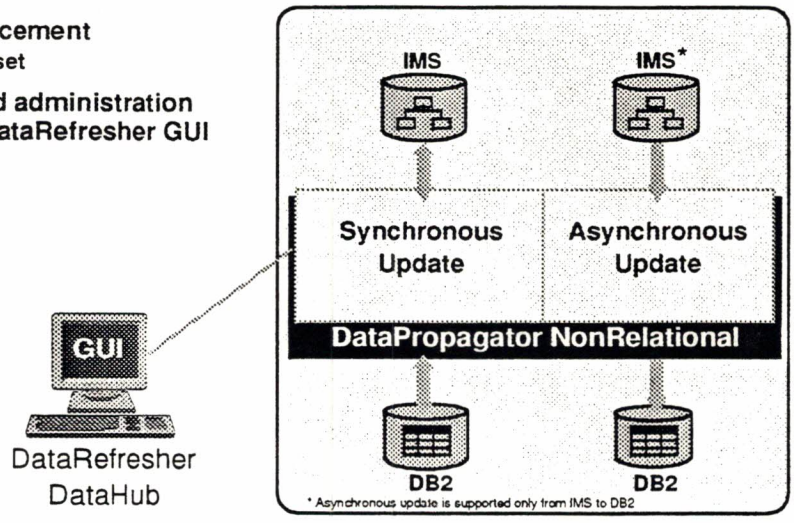


IBM Data Replicatio

IBM Software
© Copyright IBM 1995

DataPropagator NonRelational

- Enhancement
  - Subset
- Shared administration with DataRefresher GUI

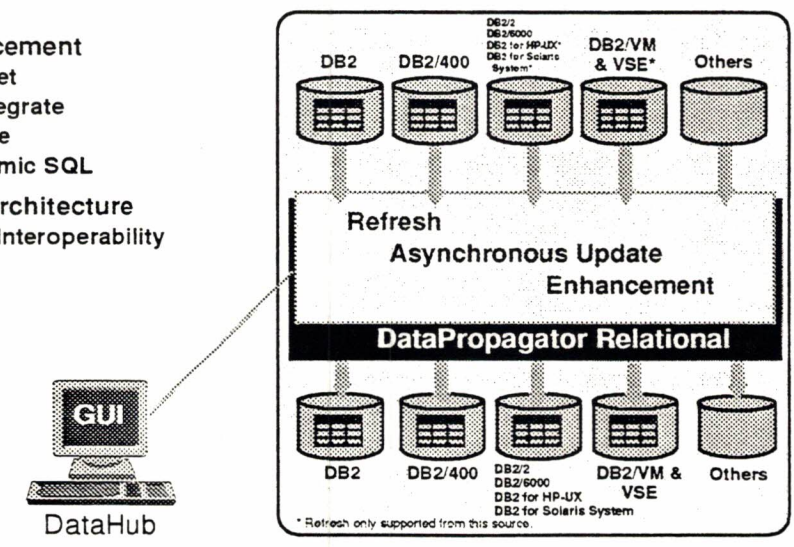


IBM Data Replication

IBM Software
© Copyright IBM 1995

DataPropagator Relational

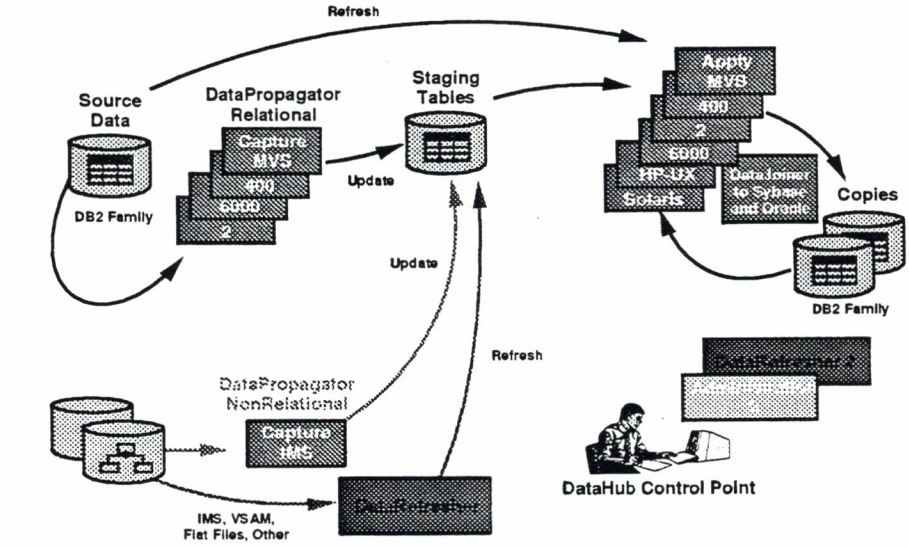
- Enhancement
  - Subset
  - Aggregrate
  - Derive
  - Dynamic SQL
- Open architecture
  - Tool Interoperability



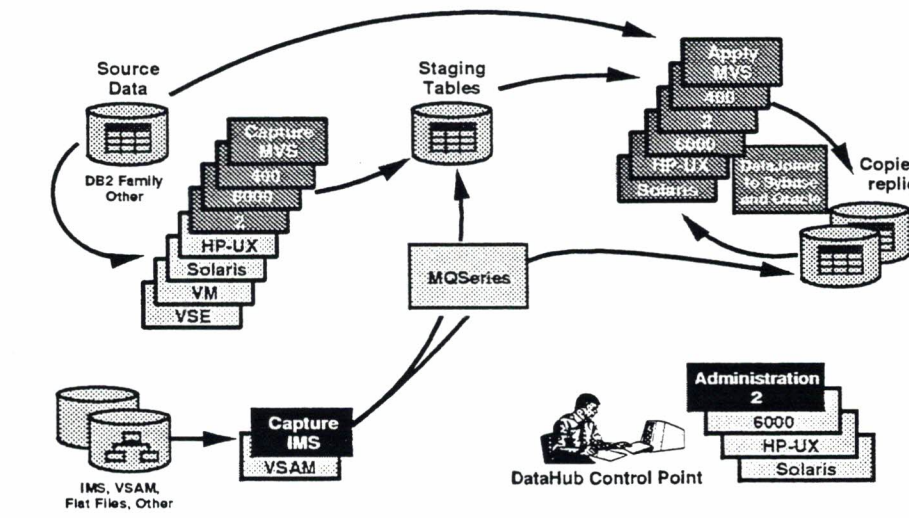
IBM Data Replication

IBM Software
© Copyright IBM 1995

Integrating IBM Data Replication Products



IBM Data Replication Futures



Database Systems Management Tools

IBM Database Systems Manage-

Manage enterprise wide multi-vendor database environments

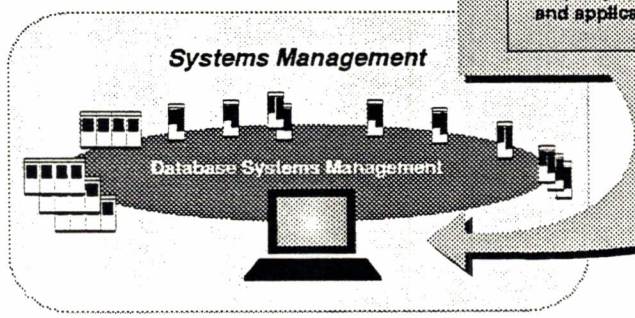
Unattended, remote database management

Scalable from small to large distributed environment

Interoperability with Systems Management

Database Systems Management Tool Categories

- DBA Tools for simplifying operational and administrative tasks
- Data Replication for full refresh or change propagation
- Advisor Tools for analyzing systems and applications
- Utilities for routine database tasks



IBM® Software

© Copyright IBM 1995

Database Systems Management

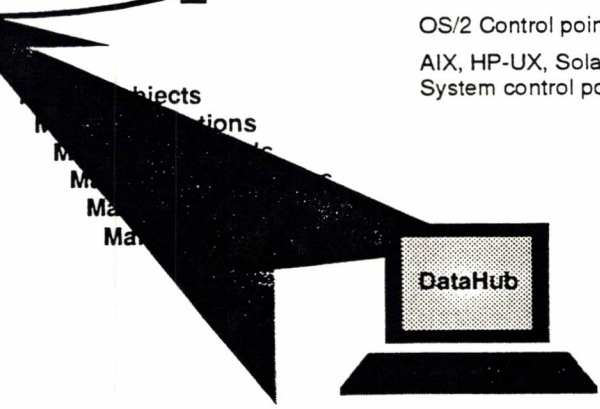
IBM Database Systems Manage-ment

DBA Tools

DataHub

OS/2 Control point

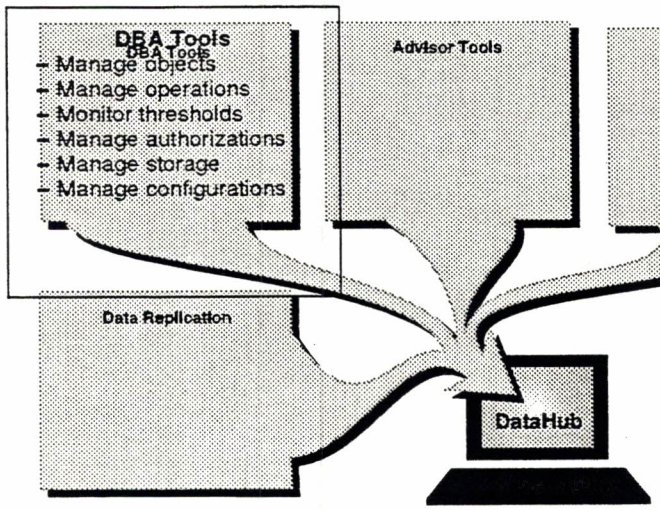
AIX, HP-UX, Solaris System control points



IBM® Software

© Copyright IBM 1995

DBA Tools



Manage multi-vendor databases, especially geographically dispersed

Facilitate expansion into client/server

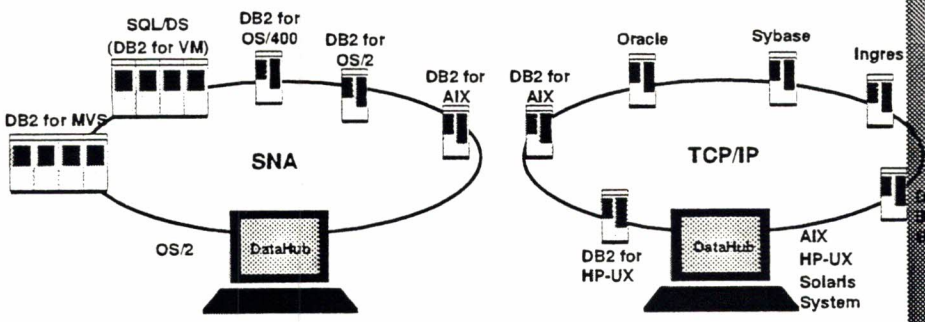
Reduce requirement for additional staff

Provide unattended remote management

IBM Database Systems Management

IBM® Software
© Copyright IBM 1995

DataHub for OS/2 and Unix

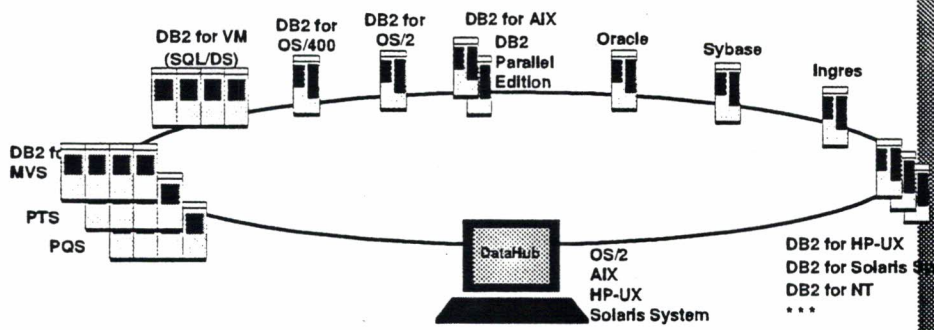


- DataHub for UNIX platforms for multi-vendor UNIX database systems management
- DataHub for OS/2 for database systems management of the DB2 family
- Scalable from small LAN to distributed enterprise
- Robust suite of tools and tool functionality
- Integration platform for tools

IBM Database Systems Management

IBM® Software
© Copyright IBM 1995

DataHub - Future Direction

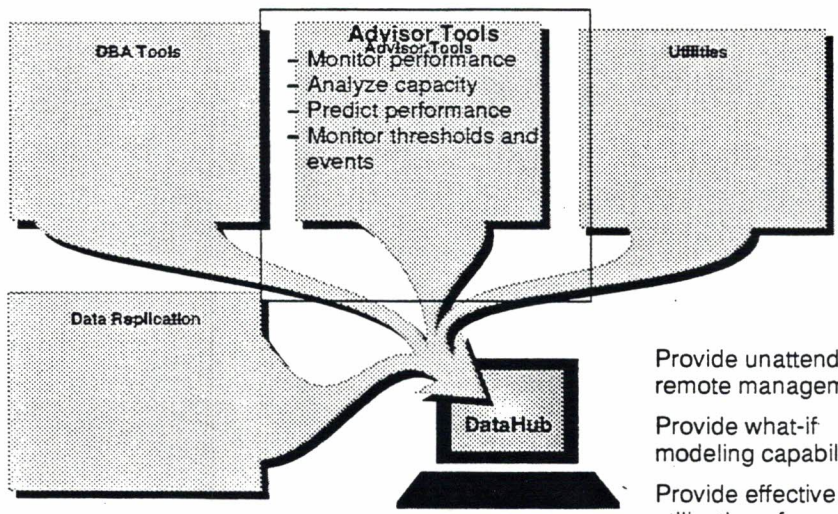


- Single control point is UNIX or OS/2
- Multi-vendor UNIX databases
- IBM DB2 family on all platforms, serial and parallel
- Robust suite of tools and tool functionality

IBM Database Systems Management

IBM Software
© Copyright IBM 1995

Advisor Tools



Provide unattended, remote management

Provide what-if modeling capability

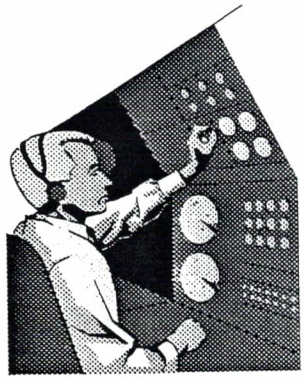
Provide effective utilization of resources

IBM Database Systems Management

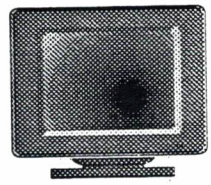
IBM Software
© Copyright IBM 1995

DB2 Performance Monitor for MVS

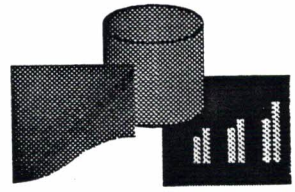
IBM Database Systems Management



- Periodic Monitoring
- Exception Monitoring
- Detail Monitoring
- In-Depth Monitoring
- DataHub-enabled



Online Monitor



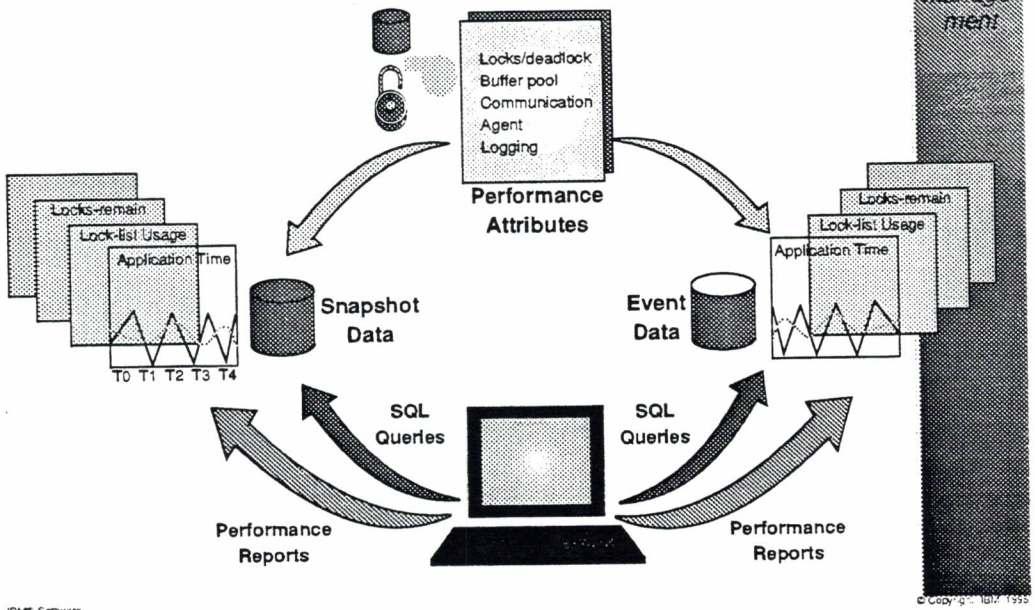
Batch Reporting

IBM® Software

© Copyright IBM 1995

DB2 Performance Monitor for OS/2 and AIX

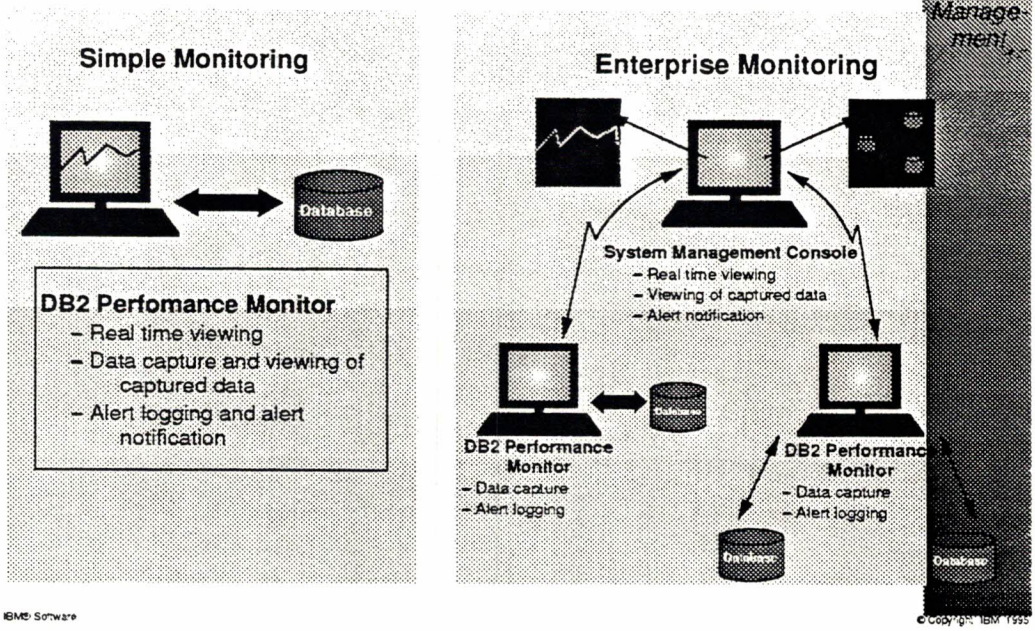
IBM Database Systems Management



IBM® Software

© Copyright IBM 1995

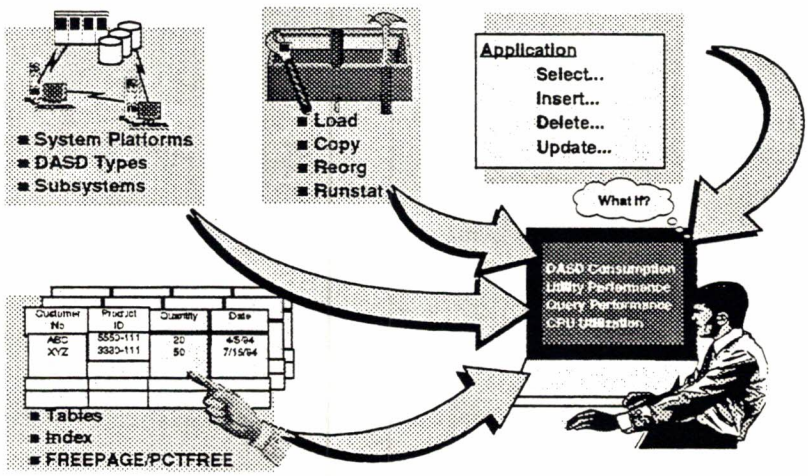
DB2 Performance Monitor for OS/2 and AIX



IBM® Software

© Copyright IBM 1995

DB2 Estimator for Windows

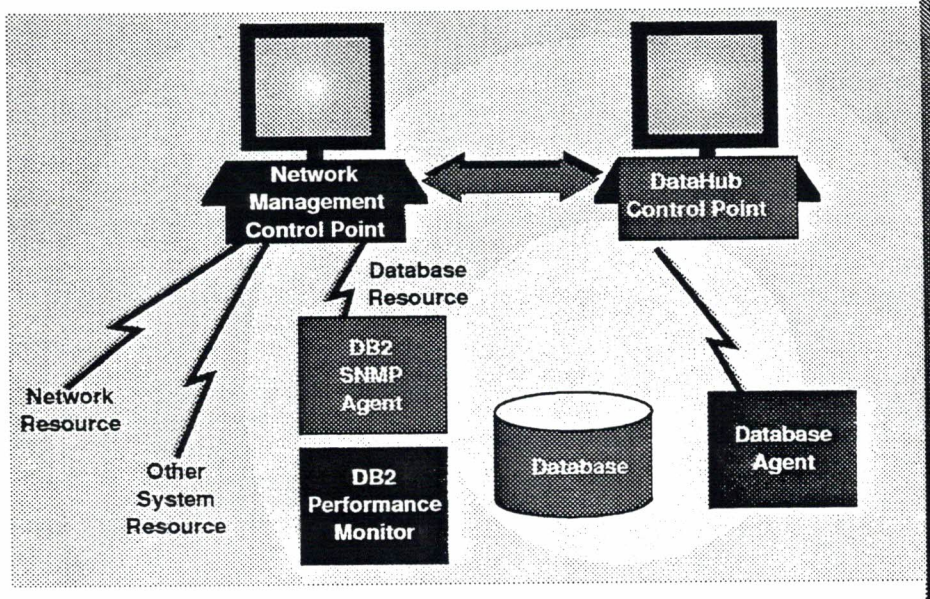


Evaluate Design Alternatives
Easy to Learn and Use
DataHub enabled

IBM® Software

© Copyright IBM 1995

Network Management Control Point

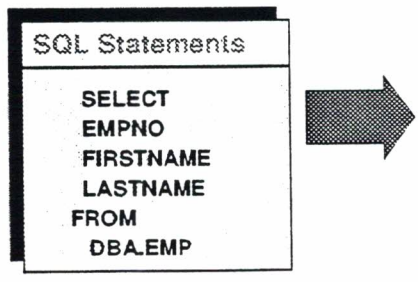


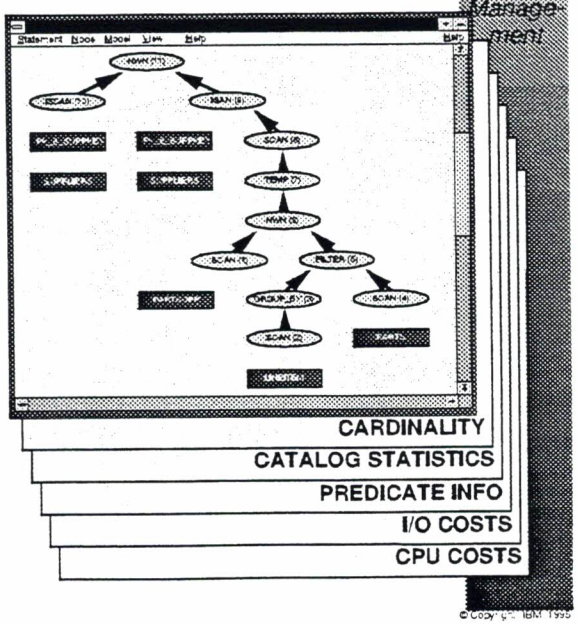
IBM Database Systems Management

IBM Software

© Copyright IBM 1995

Visual Explain





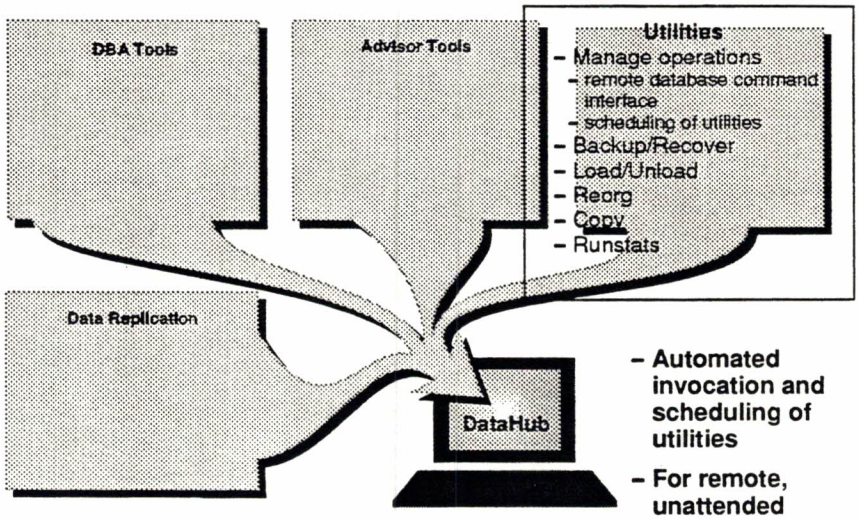
- DataHub enabled
- Graphical presentation of Explain output
- Detailed optimizer information
- What - if modeling capability for SQL
- Runs on OS/2 and on AIX

IBM Database Systems Management

IBM Software

© Copyright IBM 1995

Utilities

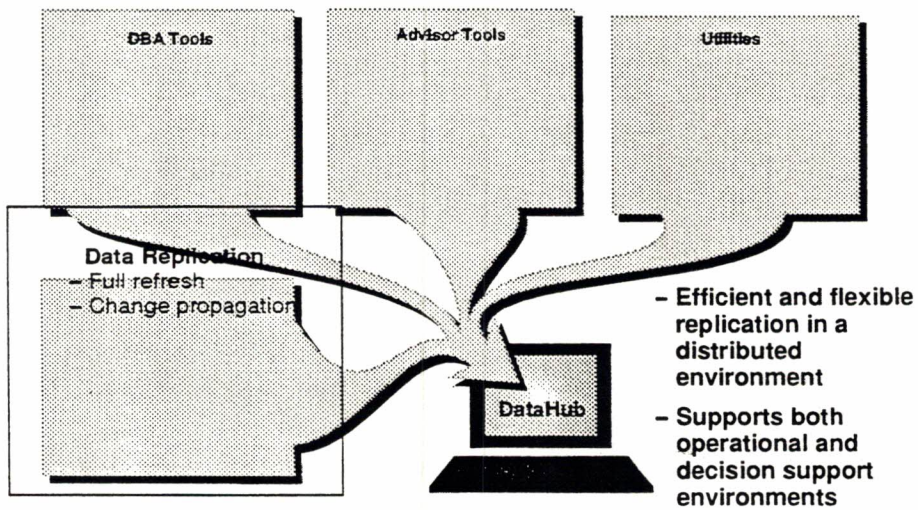


IBM Database Systems Management

IBM Software

© Copyright IBM 1995

Data Replication

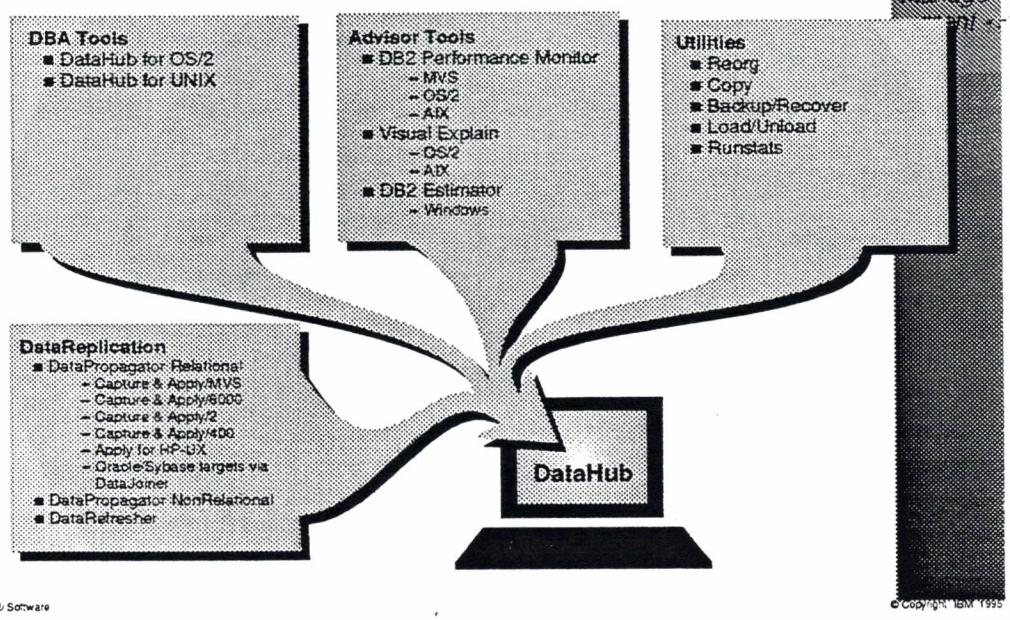


IBM Database Systems Management

IBM Software

© Copyright IBM 1995

IBM Systems Management Products



DB2 Sources of Information

- Compuserve
  - IBMDB2 Forum
    - ► Section 1 - DB2/2 Questions and Answers
    - ► Section 2 - DB2/6000 Questions and Answers
    - ► Library 1 - DB2/2 Information
    - ► Library 2 - DB2/6000 Information
  - IBMLINK/TALKLINK
  - OS2BBS
    - ► DB26000 FORUM - Questions and Answers
    - ► DB2OS2 FORUM - Questions and Answers
- Internet
  - COMP.DATABASES
    - ► Monitored and questions about DB2/2 and DB2/6000 are answered
- 1-800-992-4777 DB2/2 Questions (PMR)
- 1-800-237-5511 DB2/6000 Questions (PMR)

CS 0903
Introduction to APPC and APPN

Dave Lutz
IBM

Agenda

- Definitions: What's the difference between APPC and APPN?
- Terminology: What's up with those strange words and acronyms?
- Benefits: What can APPC and APPN do for you and your network?
- Connectivity: Who's got APPC and APPN implementations for you to use?

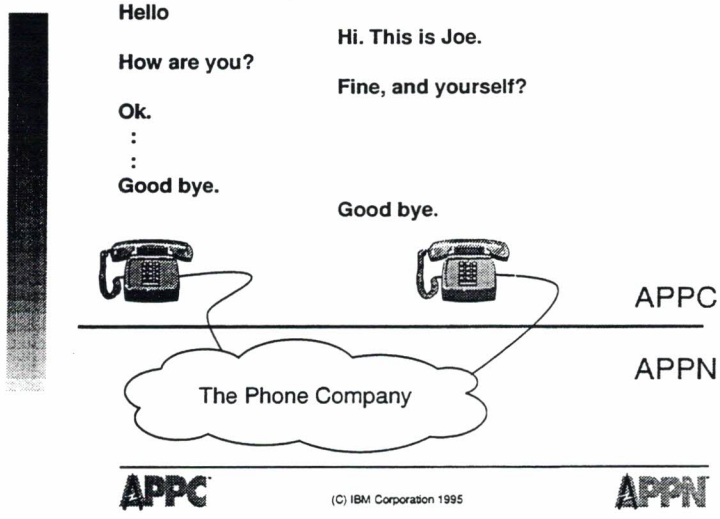
APPC (C) IBM Corporation 1995 APPN

What All Those Letters Stand For

- APPC: Advanced Program-to-Program Communication
- APPN: Advanced Peer-to-Peer Networking

APPC (C) IBM Corporation 1995 APPN

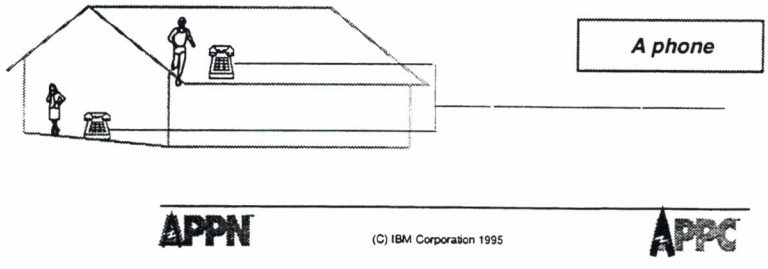
APPC and APPN



What Is an LU?

Logical Unit - *simply the SNA communications software in a computer*

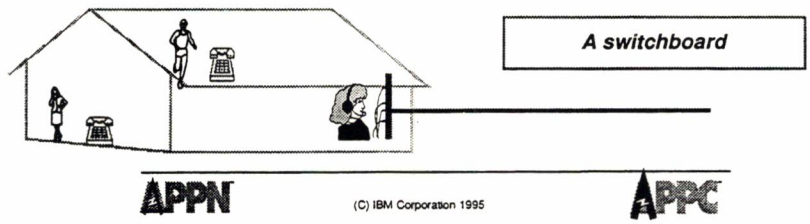
- Acts upon requests from a user's program
- Reacts to communication adapters
- Manages the exchange of data between application programs



What Is LU 6.2?

Logical Unit 6.2 - *most advanced form of LU best known as* ***APPC***

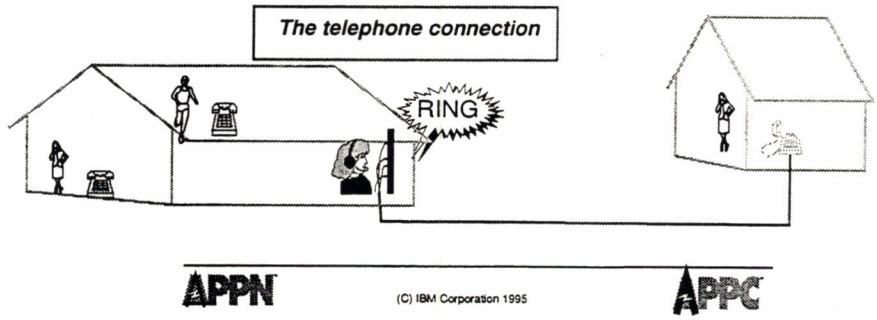
- Enables peer communications between computers
- Creates a transparent environment
  - Provides communication services to all APPC applications on the computer
  - Frees applications from network specific details
  - Increases processing power for PC users without the constraints of mainframe-based slave devices



What Is a Session?

Session - *logical connection between two LUs*

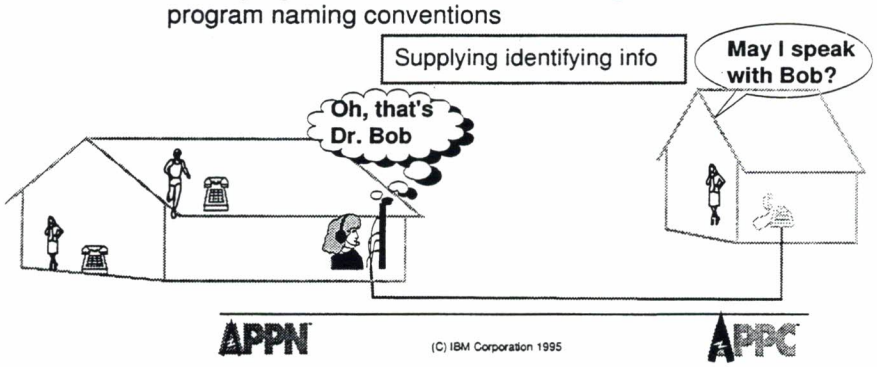
- Manages the exchange of data between LUs
  - Quantity of data exchanged
  - Data security
  - Traffic congestion
  - Network routing



What Is a TP Name?

TP Name - *"Transaction Program" Name*

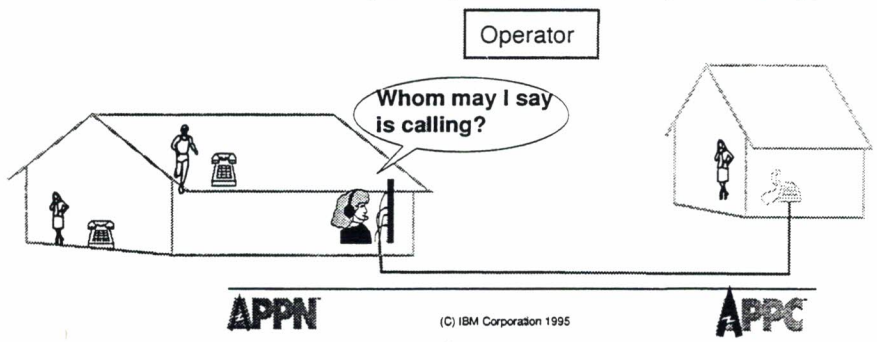
- Name sent by the client program that identifies a partner program
- Client program remains unaware of target node's program naming conventions



Attach Manager

Attach Manager - *Manages every incoming request for programs on the computer*

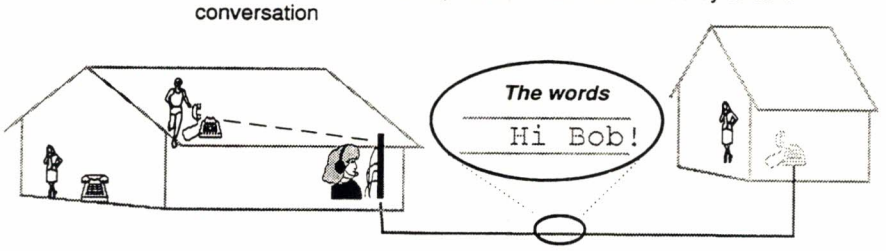
- Uses the TP Name to connect to the correct target program
- Performs compatibility checks (for example: security)



What Is a Conversation?

Conversation - *data exchanged between two programs*

- Only one conversation per session at a time
- A session generally remains active between conversations
  - A conversation is short compared to a session
  - After a conversation ends, a session can be reused by another conversation



APPN (C) IBM Corporation 1995 APPC

Why Use APPC?

- For its built-in features for client/server applications
- For its portability
- For its performance
- For its capability in both LAN and WAN environments
- For its connectivity
  - IBM platforms
  - Non-IBM platforms

APPC (C) IBM Corporation 1995 APPN

APPC Built-In Features

- Functions important for client/server development
  - Class of Service (COS)
  - Security
  - Synchronization
  - Error recovery
- Benefits
  - Reduced development time
  - Lower development costs

APPC (C) IBM Corporation 1995 APPN

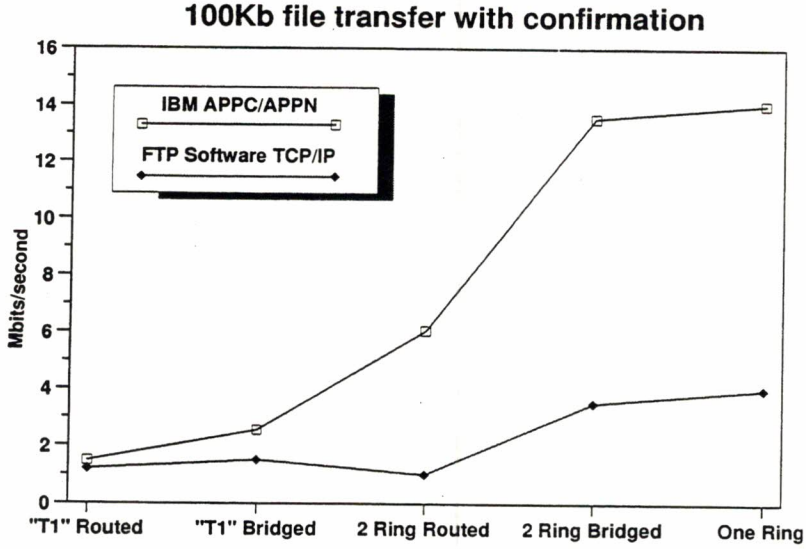
Portability

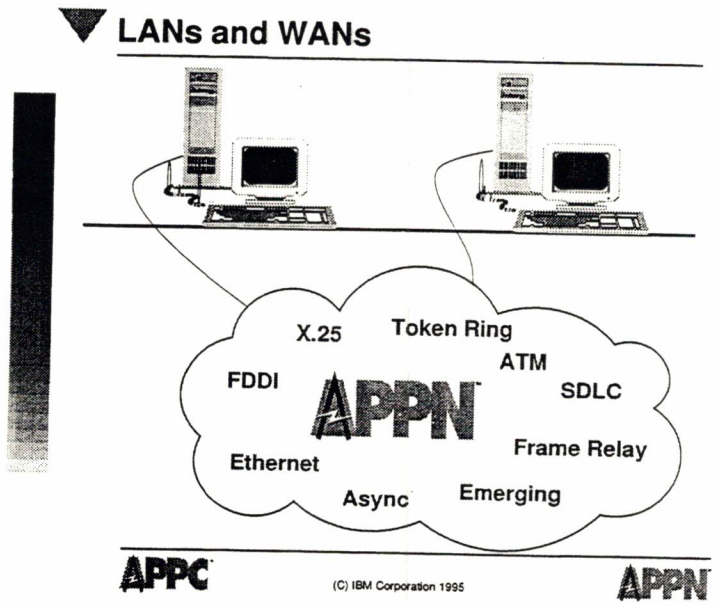
Common Programming Interface for Communications (CPI-C)

- Programming interface for APPC
- Consistency across platforms and languages
- Portability of applications
- Portability of skills
- Communication with native APPC programs

APPC (C) IBM Corporation 1995 APPN

Vedacom's OS/2 Performance Results





IBM APPC Platforms

- AS/400
- CICS
- DOS
- DPPX/370
- IMS
- MVS
- OS/2
- RISC System/6000
- Series/1
- System/36
- System/38
- System/88
- TPF
- VM
- VTAM
- 4680/4690 Store System

APPC (C) IBM Corporation 1995 APPN

Non-IBM APPC Platforms

- Amdahl
- Apertus Technologies, Inc.
- Apple Computer, Inc.
- AT&T/NCR
- Attachmate
- Brixton Systems, Inc.
- Bull Information Systems
- CLEO Communications
- Compsoft Services
- Computone
- Data Connection Ltd.
- Data General Corp.
- Digital Equipment Corp.
- Eicon Technology, Inc.
- Harris
- Hewlett-Packard Co.
- Hitachi
- Insession
- Microsoft
- NetSoft
- Novell, Inc.
- OpenConnect Systems
- Orion Group
- Rabbit Software, Corp.
- Stratus Computer Inc.
- Sun Microsystems, Inc.
- Tandem Computers, Inc.
- Tangram Systems Corp.
- Texas Instruments
- Unihorn
- Unisync
- Unisys
- Wall Data
- Wang Laboratories, Inc.

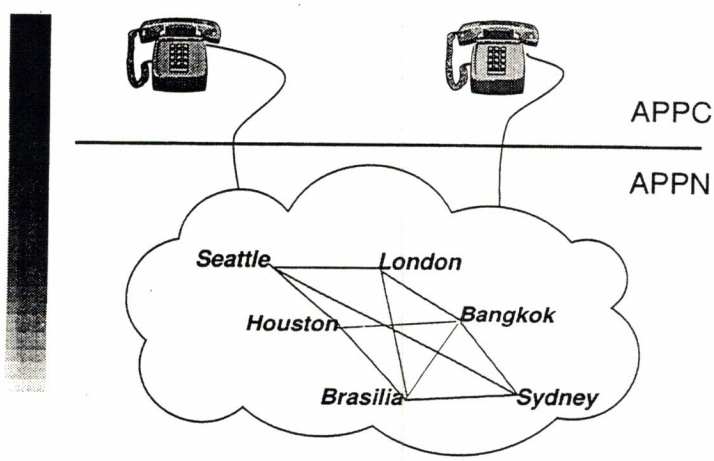
APPC (C) IBM Corporation 1995 APPN

Vendors With APPC Applications

- Adelie Corp.
- ADP
- Affinity Financial Group
- ALE Systems
- Alldata Service GmbH
- American Management Systems, Inc.
- American Software USA
- Andersen Consulting
- Annatek Systems, Inc.
- Applied Computer Technology
- Apluse Tech. Comm. Co., Ltd.
- Aristacom International Inc.
- Arkansas Systems, Inc.
- Argo Data
- Attachmate
- Automatic Data Processing
- Automated Financial Systems, Inc.
- Bank of Boston
- Business Partner Solutions
- C&K Software Inc.
- Candle Corporation
- CCS Commercial Credit Systems
- Cleo Communications
- Citisystems, Inc.
- COL Consulting, Ltd.
- Computer Associates International, Inc.
- Computer Corporation of America
- Computer Science Corp. CIS
- Compuware Corporation
- Core Technology Corporation
- Corrigan and Colville, Inc.
- Danware Data A/S
- DataBase Architects, Inc.
- Data Sciences GmbH
- David Goodenough & Associates, Ltd.
- Dun & Bradstreet Software Services, Inc.
- EDS Corp.
- Enghouse Systems Ltd.
- Enterprise Data, Inc.
- FileTek Inc.
- German Telephone Service
- Heuristics, Inc.
- Hogan Systems Inc.
- I & G Software Associates, Inc.
- Infinite Options, Inc.
- Inform GmbH Co KG
- Information Builders
- Internal Systems, Inc.
- International TeleManagement Corp. (ITM)
- Intersolv, Inc.
- and many more

APPC (C) IBM Corporation 1995 APPN

APPC and APPN



APPC (C) IBM Corporation 1995 APPN

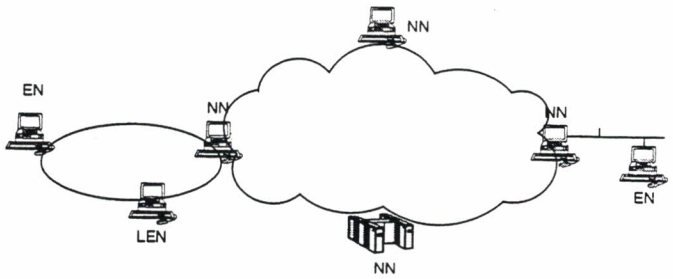
What Does APPN Do for Me?

"Plug and work" networking

- Reduces costs and increases productivity
  - Reduces network configuration work drastically
  - Finds resources automatically
  - Picks 'smartest' route to destination
- Works for both IBM and non-IBM platforms
  - Open architecture
  - Widely available

APPC (C) IBM Corporation 1995 APPN

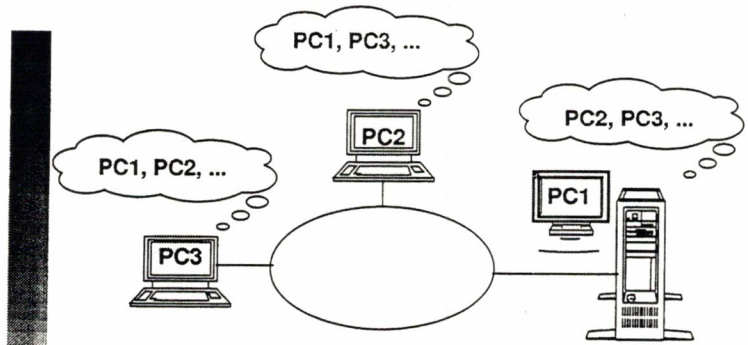
APPN Network Participants



- **Network Node (NN): provides directory/topology services**
- **End Node (EN): uses network node services**
- **Low Entry Networking Node (LEN): communicates peer to peer**

APPN (C) IBM Corporation 1995 APPC

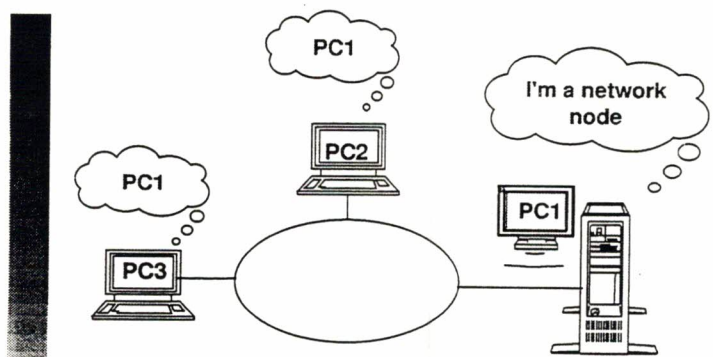
APPC Configuration Without APPN



For peer communications, each machine must have definitions for every other machine.

APPC (C) IBM Corporation 1995 APPN

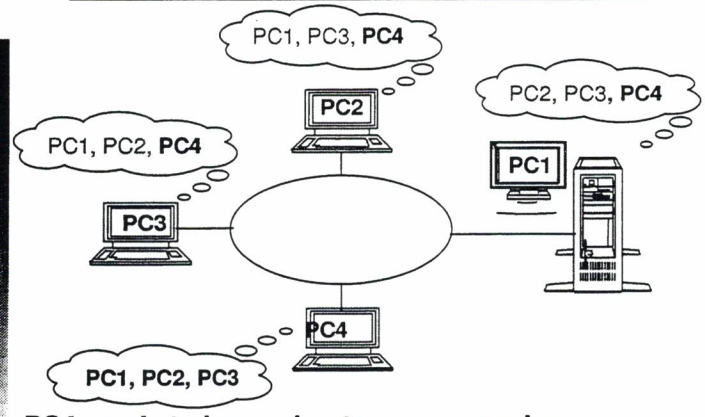
APPC Configuration With APPN



Each machine only knows about its network node

APPC (C) IBM Corporation 1995 APPN

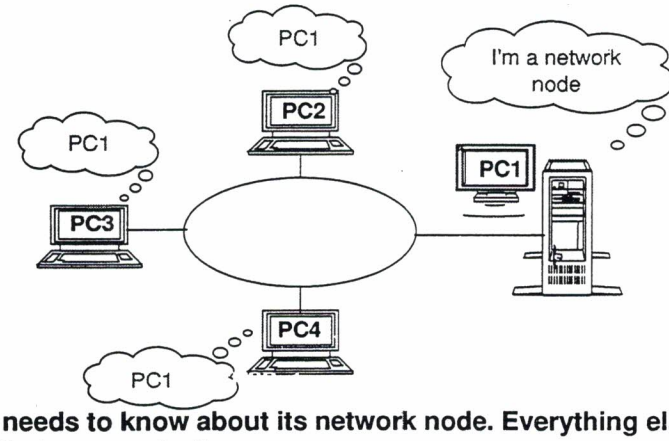
Network Growth Without APPN



PC4 needs to know about everyone and everyone needs to know about PC4

APPC (C) IBM Corporation 1995 APPN

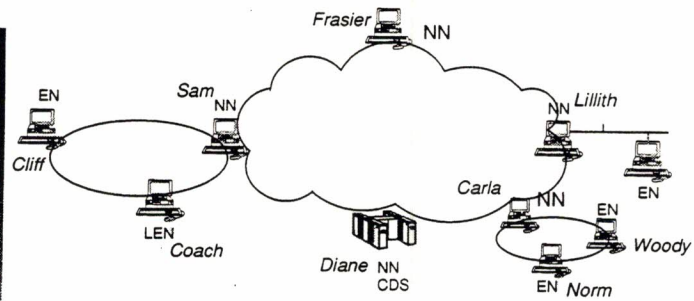
Network Growth With APPN



PC4 needs to know about its network node. Everything else is handled automatically.

APPC (C) IBM Corporation 1995 APPN

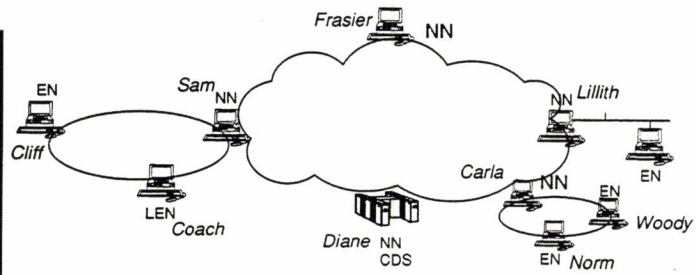
APPN Directory



- NNs have directory of their ENs and other NNs
- Dynamically locate network resources on other NNs
- Use Central Directory Server (CDS)

APPC (C) IBM Corporation 1995 APPN

APPN Topology



- Recognize and record link characteristics
- Exchange topology information with other NNs
- Retain information across outages
- Select smartest route based on current conditions of network
- Use 'Connection networks'

APPC (C) IBM Corporation 1995 APPN

Open Architectures

- APPN Implementers Workshop (AIW)
- EN and NN architecture specifications available
- EN and NN technology agreement
- NN source code licensing

APPC (C) IBM Corporation 1995 APPN

APPN is Supported in the Industry

- IBM
  - AIX SNA Server/6000
  - AS/400
  - CM/2
  - DPPX/370
  - NetDA/2
  - NetView
  - System/36
  - VTAM
  - 2210 Nways Router
  - 2217 Mulitprotocol Concentrator
  - 3172 Interconnect Controller
  - 3174
  - 3746 Nways Controller
  - 6611 Network Processor
  - 8250
  - 8260
- Non-IBM
  - Advanced Computer Comm.
  - Apple
  - Apertus Technologies, Inc.
  - * Ascom Timeplex
  - Attachmate
  - Bay Networks
  - Brixton Systems
  - Bytex
  - Cabletron Systems
  - * Cisco
  - * CrossComm
  - Data Connection Ltd.
  - Eicon Technology
  - Hewlett-Packard
  - Insession
  - Hitachi Data Systems
  - Network Systems Corp.
  - * N.E.T.
  - Northern Telecom
  - Novell
  - * Proteon
  - Siemens-Nixdorf
  - Sync Research
  - Tandem
  - Ungermann-Bass
  - Unisys
  - Wall Data
  - 3Com

★ - have licensed APPN NN code from IBM

APPC (C) IBM Corporation 1995 APPN

APPC and APPN Benefits

- Robust networking capabilities
- Ideal base for new client/server applications
- Reduced network definitions and increased connectivity
- Wide range of vendors and platforms - open and available

APPN (C) IBM Corporation 1995 APPC

APPC and APPN Online Information

- WWW:
  http://www.raleigh.ibm.com/aac/aachome.htm
- FTP site: anonymous ftp to
  networking.raleigh.ibm.com (192.35.236.4)
- CompuServe: GO APPC
- MKTTOOLS (IBM internal)

APPC (C) IBM Corporation 1995 APPN

More information about APPC and APPN

- The Best of APPC, APPN and CPI-C CD-ROM (order number SK2T-2013)
- The APPC Connection
- The APPC/APPN Technical Conference World Tour '95
  - Chicago May 22-25
  - London September 4-7
  - Sydney October 16-19

APPC (C) IBM Corporation 1995 APPN

LAN 0904
What's New in IBM TCP/IP for OS/2?

John Conner
IBM

OS/2 Warp Connect, the newest addition to IBM's PC operating system family, now ships with a TCP/IP for OS/2 Version 3 that lets users simultaneously connect to local/wide area networks and to the Internet with features that enhance web browsing, mail and file transfer. This presentation will describe what existing and new IBM TCP/IP for OS/2 features have been included in OS/2 Warp Connect. It will also discuss what other TCP/IP for OS/2 features and functions are available from IBM. The focus here will be on how these features and functions are packaged, as well as on how they work together. Future TCP/IP trends and directions will also be discussed.

TCP/IP FUNCTIONALITY IN OS/2 WARP CONNECT

The IBM TCP/IP for OS/2 Version 3 function in OS/2 Warp Connect combines all of the 'commercial' TCP/IP functionality of the popular IBM TCP/IP Version 2.0 for OS/2 product with the ease-of-use features of the Internet Connection for OS/2 found in the OS/2 Warp BonusPak.

Some of the more popular TCP/IP features and functions now included are:

- IBM WebExplorer: is a World-Wide Web (WWW) browser for OS/2. This program supplies a graphically rich, interactive interface to the fascinating world of the Internet. Users of the WebExplorer can jump from one Internet location to another by simply selecting highlighted text or images in the main window.

- UltiMedia Mail/2 Lite: provides an easy-to-use interface for creating, sending and receiving, and storing electronic mail. Ultimedia Mail/2 Lite supports MIME protocol which enables multimedia mail.

- Gopher Client: provides a graphical, user-friendly interface to information from the Internet or the user's LAN.

- NewsReader/2: provides a graphical interface to LAN or Internet connected news servers.

- TelnetPM: allows you to login to other computers on the Internet or the user's LAN. TelnetPM provides terminal access to programs and information on other computers. TelnetPM supports VT100 and VT220 emulation. A telnet server is also provided, to allow terminal access to an OS/2 computer from authorized users on other computers.

- PMANT: allows access to information and programs located on Internet or LAN connected hosts that support 3270 sessions, such as VM or MVS, using TN3270 emulation. PMANT is an OS/2 Presentation Manager application. TN5250 is available for terminal access to AS/400 computers.

- FTPPM: allows you to transfer files easily between your computer and other computers on the Internet or LAN. If you are authorized access to a given computer, you can manipulate files on that computer as if it were your own. FTPPM is an OS/2 Presentation Manager application. An FTP server is also provided so that authorized users on other computers on the network can access files on OS/2 computers.

• LPR/LPD: provides users the ability to print files on TCP/IP print servers on the network, which may be AIX, UNIX, OS/2, or other types of computers. The LPD server which is also provided allows any OS/2 machine to function as a TCP/IP print server.

• SNMP: allows network management by elements, such as gateways, routers, and hosts. This function provides a means of communication between network elements regarding network resources. TCP/IP Version 3 provides SNMP agent and manager functions, and the agent is extensible: new subagents can be created to manage additional computer functions, using the DPI 2.0 interface.

• PPP: Point-to-Point Protocol support has been added to provide users with improved network accessibility. SLIP dial support has also been enhanced.

PACKAGING/COMPATIBILITY

When the IBM TCP/IP Version 2 for OS/2 TCP/IP product was made available in August of 1993, it was packaged in a number of separate product 'kits' to allow customers the ability to pick and choose what applications they required. The 'kits' that can now be considered to be included in Warp Connect are the TCP/IP for OS/2 Base Kit and DOS/Windows Access Kit.

The TCP/IP support in OS/2 Warp Connect remains compatible with the following existing TCP/IP Version 2 for OS/2 Kits:

- Network File System
- Domain Name Server
- Extended Networking
- Programmer's Toolkit
- X Window System Server
- X Window System Client
- OSF/Motif

The function found in the TCP/IP Version 2 for OS/2 NetBIOS Kit is not supported on OS/2 Warp Connect. It has been replaced by the RFC NetBIOS over TCP/IP function found in LAN Server 4.0 and in the LAN Requester feature found in OS/2 Warp Connect.

The TCP/IP Version 2 for OS/2 Application and Ultimedia Mail/2 Kits are not supported on OS/2 Warp Connect. The function found in the Application kit is included in OS/2 Warp Connect. The Ultimedia Mail/2 kit has been replaced by the IBM Workgroup product.

LAN 0906
Configuring Multiple Protocols in a Workstation

Laura Jean Knapp
IBM

IBM Networking

Connecting people to information through technology

LAN Adapter and Protocols

OS/2 Workstation Configuration

Laura Knapp
Senior Systems Advisor
IBM Networking

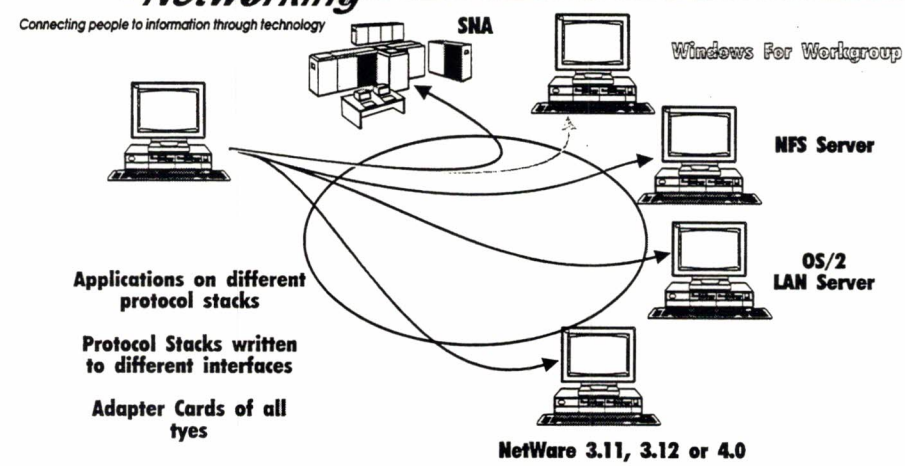
1-919-543-7049
70534,3510
LJKNAPP@RALVM12.VNET.IBM.COM

IBMNetworking © Copyright IBM Corporation 1993,1994, 1995 ADPT001A

IBM Networking

Connecting people to information through technology

What are we trying to do !



IBMNetworking © Copyright IBM Corporation 1993, 1994, 1995 ADPT002A

IBM Networking

Connecting people to information through technology

Issues for Adapter Developers

How to write a protocol stack independent of the adapter?

How to get multiple protocols using a single adapter card?

How to easily develop device drivers for adapters?

802.2

Native

NDIS

ODI

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

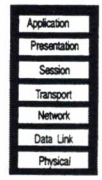
Connecting people to information through technology

NDIS/ODI/802.2 - What's the Difference !

NDIS
Network Driver Interface Specification

- Joint 3-Com/Microsoft effort
- Standard interface for adapter drivers and protocol drivers
- Separates protocol handling from hardware
- Dependent on Network protocols

802.2



- Defines interfaces within system
- End-end between two communicating systems
- OSI Standard
- Higher layer written to 802.2 interface and 802.2 written to adapter interface

ODI
Open Data-Link Standard

- Novell/Apple developed
- Protocol independence at the data link layer
- Allows multiple protocols to share a single LAN Adapter
- Independent of Network protocols
- LAN adapter driver written to ODI specifications

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

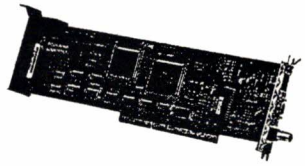
NDIS/ODI/802.2 - OSI Reference

Protocol

I/F

Device Driver

I/F

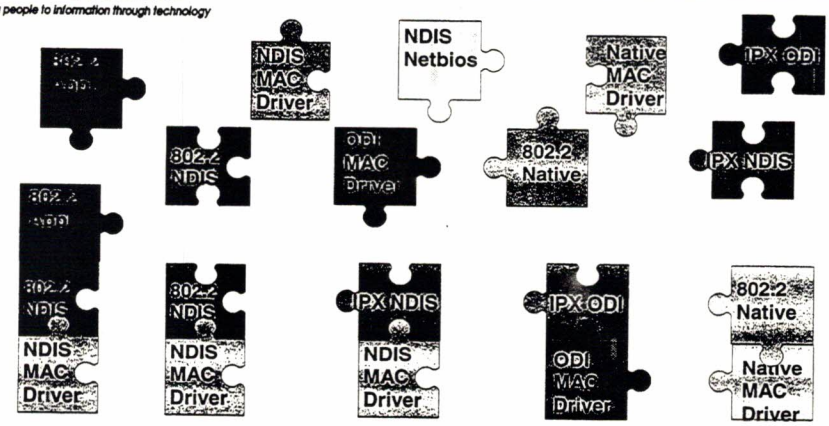


© Copyright IBM Corporation 1993, 1994, 1995

NETWORK ACCESS

Connecting people to information through technology

NDIS/ODI/802.2 -Puzzeled ?

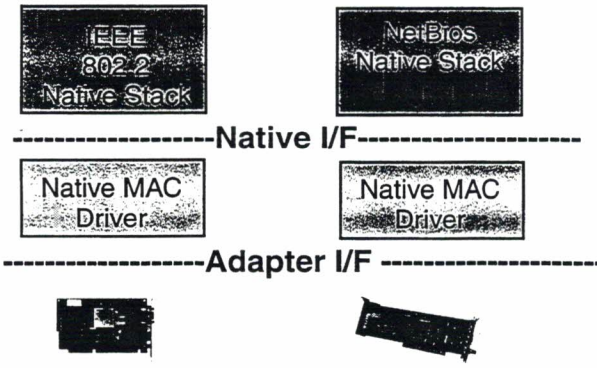


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Native (vendor proprietary) Interface



Each vendor had own specification for writing adapter drivers, protocol stacks etc.

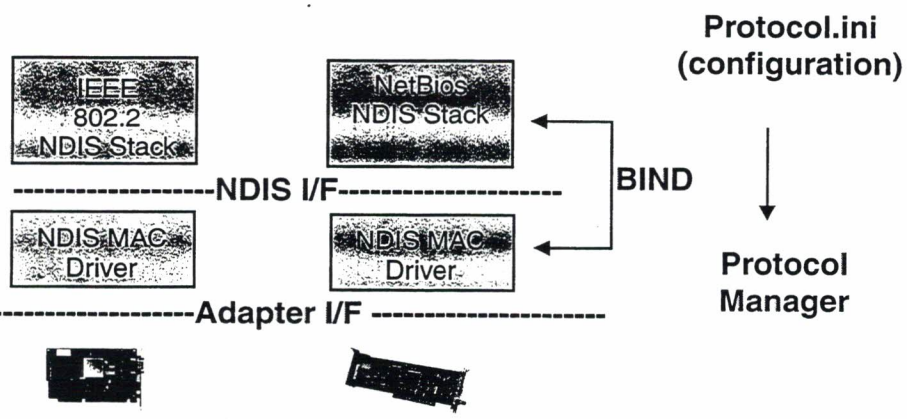
Paramters passed as fields with each vendors implementation

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT007A

IBM Networking

Connecting people to information through technology

NDIS Interface

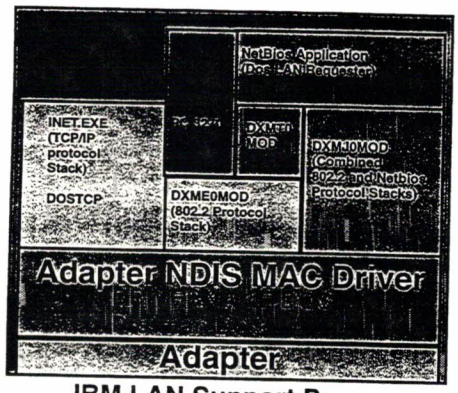


IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT007B

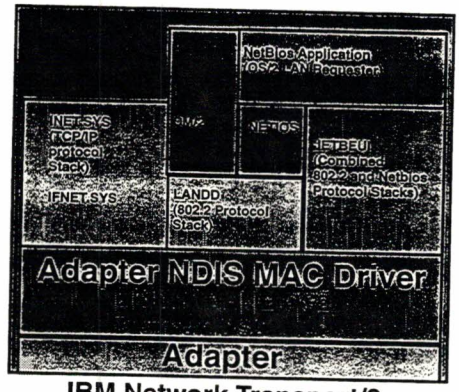
IBM NDIS Implementation

IBM Networking

Connecting people to information through technology



IBM LAN Support Program
TCP/IP V 2.1 DOS



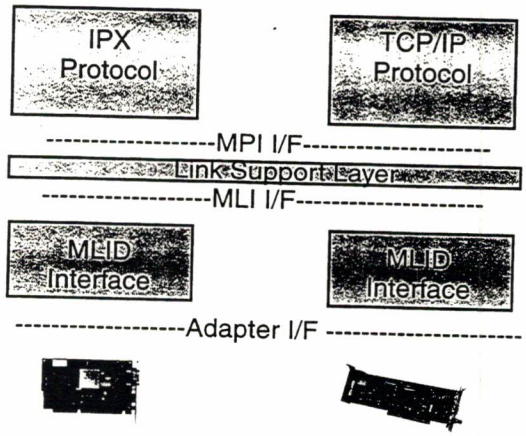
IBM Network Transport/2
TCP/IP for OS/2

© Copyright IBM Corporation 1993, 1994, 1995

ODI Interface

IBM Networking

Connecting people to information through technology



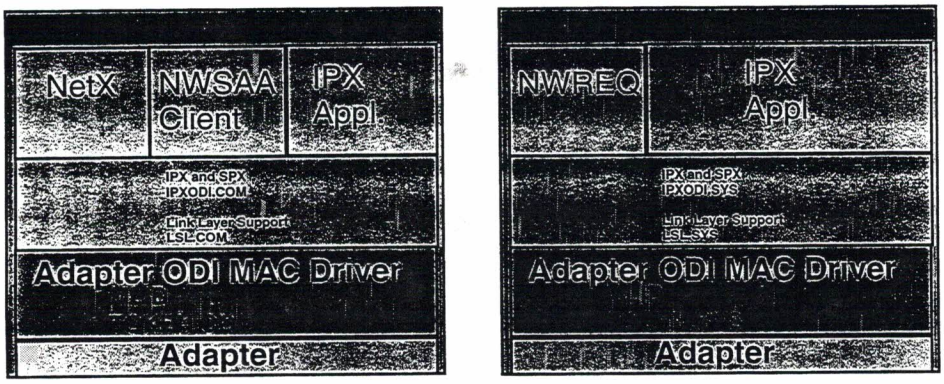
Parameters are passed using the 'NET.CFG' file

© Copyright IBM Corporation 1993, 1994, 1995

ODI Implementation

IBM Networking

Connecting people to information through technology



Novell Client Code for DOS

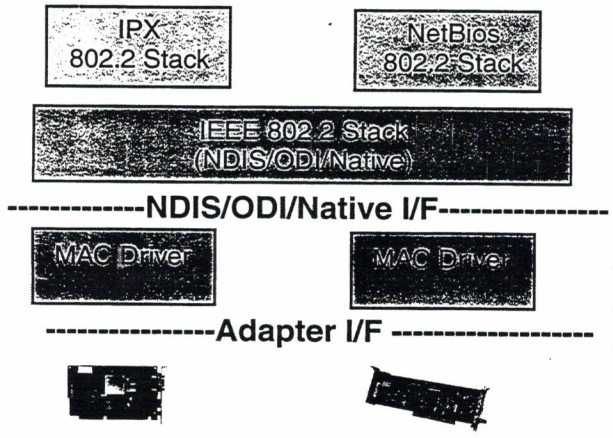
Novell Client Code for OS/2

© Copyright IBM Corporation 1993, 1994, 1995

802.2 Interface

IBM Networking

Connecting people to information through technology



Novell is moving to this configuration. Bypassing the 802.2 stack is non-standard and hampers interconnectivity.

IBM has always backed this mechanism to provide support for multiple protocols.

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Operating System and Driver Interface

	Native	NDIS	ODI
DOS/Windows (Lan Support Program)	DXMC0MOD.SYS	DXME0MOD.SYS IBM16TR.DOS	IPXODI.COM TOKEN.COM
OS/2 (NTS/2)		LANDD.OS2 IBMTOK.OS2	IPXODI.SYS TOKEN.SYS
PCMCIA (Ships with Adapter)		IBMTOKCS.OS2 IBMTOKCS.SYS	TOKENCS.COM TOKENCS.SYS

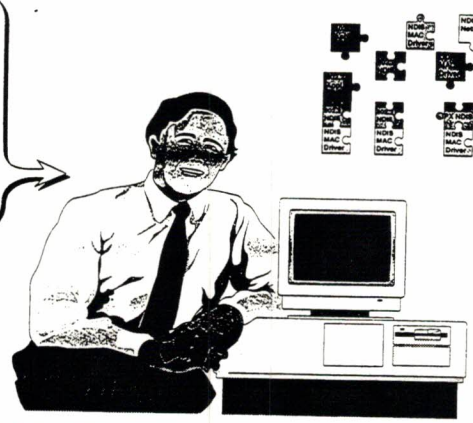
© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

How do I configure my system?

I need to get NetWare Clients, SNA (802.2), Netbios and TCP/IP all running on the same system.... is this an impossible dream !!

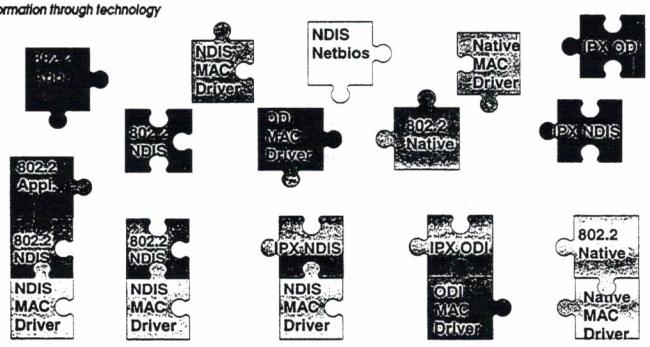


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Protocol Interfaces



NDIS
ODI
ASI (Native LAN Support Program)

Packet Driver
Direct
DEC DLL, etc

© Copyright IBM Corporation 1993, 1994, 1995

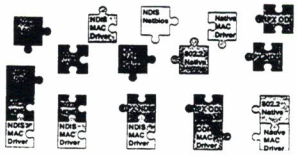
IBM Networking

Connecting people to information through technology

Coexistence Options

LANSUP

Takes ODI requests and allows them to flow over 802.2
Novell is moving to utilizing 802.2



ODINSUP

Takes NDIS protocol stacks and allows them to flow over ODI
Used when ODI MAC adapter driver is loaded and NDIS protocols are to be used
Use if primary work is ODI (Novell)

ODI2NDI

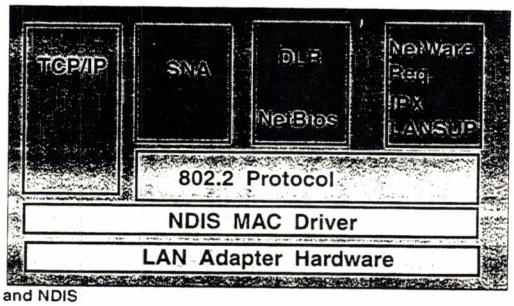
Takes ODI protocol stacks and allows them to flow over NDI
Used when the NDIS MAC adapter driver is loaded and ODI protocols are to be used
Use if primary work is NDIS (like SNA, WFW, TCP/IP)

© Copyright IBM Corporation 1993, 1994, 1995

Mixed Environment - LANSUP

IBM Networking

Connecting people to information through technology



LANSUP
- Takes ODI requests and allows them to flow over 802.2
- Novell is moving to utilizing 802.2

ODINSUP
- Takes NDIS protocol stacks and allows them to flow over ODI
- Used when ODI MAC adapter driver is loaded and NDIS protocols are to be used
- Use if primary work is ODI (Novell)

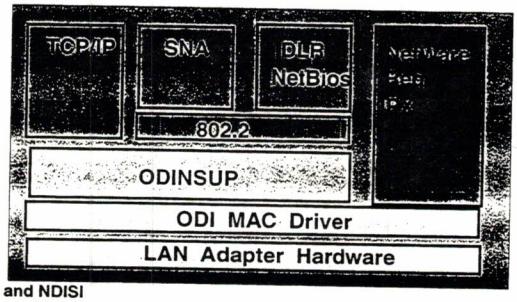
ODI2NDI
- Takes ODI protocol stacks and allows them to flow over NDIS
- Used when the NDIS MAC adapter driver is loaded and ODI protocols are to be used
- Use if primary work is NDIS (like WFW, TCP/IP, SNA)

© Copyright IBM Corporation 1993, 1994, 1995

Mixed Environment - ODINSUP

IBM Networking

Connecting people to information through technology



LANSUP
- Takes ODI requests and allows them to flow over 802.2
- Novell is moving to utilizing 802.2

ODINSUP
- Takes NDIS protocol stacks and allows them to flow over ODI
- Used when ODI MAC adapter driver is loaded and NDISI protocols are to be used
- Use if primary work is ODI (Novell)

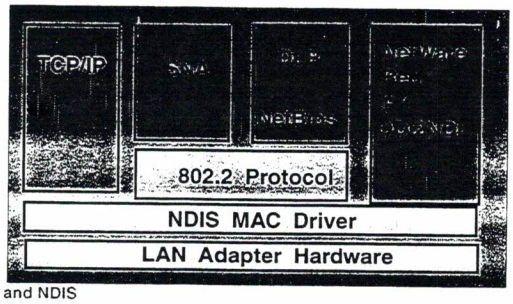
ODI2NDI
- Takes ODI protocol stacks and allows them to flow over NDIS
- Used when the NDIS MAC adapter driver is loaded and ODI protocols are to be used
- Use if primary work is NDIS (like WFW, SNA, TCP/IP)

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Mixed Environment - ODI2NDI



LANSUP
- Takes ODI requests and allows them to flow over 802.2
- Novell is moving to utilizing 802.2

ODINSUP
- Takes NDIS protocol stacks and allows them to flow over ODI
- Used when ODI MAC adapter driver is loaded and NDIS protocols are to be used
- Use if primary work is ODI (Novell)

ODI2NDI
- **Takes ODI protocol stacks and allows them to flow over NDIS**
- **Used when the NDIS MAC adapter driver is loaded and ODI protocols are to be used**
- **Use if primary work is NDIS (WFW, TCP/IP, etc)**

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

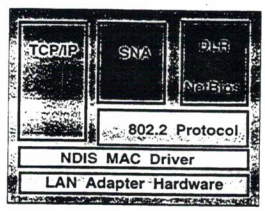
Connecting people to information through technology

All NDIS Applications

TCP/IP, Communications Manager/2, OS/2

CONFIG.SYS

```
DEVICE=C:\IBMCOM\PRTOOCOL\LANPDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
DEVICE=C:\C:\MMOS2\ADSHDD.SYS
DEVICE=C:\MMOS2\SMVDD.SYS
RUN=C:\IBMCOM\PRTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
DEVICE=C:\IBMCOM\PROTOCOL\INET.SYS
DEVICE=C:\IBMCOM\PROTOCOL\IFNDIS.SYS
DEVICE=C:\IBMCOM\MACS\IBMTOK.OS2
REM *** SERIES OF SET COMMANDS FOR TCP ****
RUN=C:\OSTCP200\BIN\CNTRL.EXE
IFS=C:\OSTCP200\BIN\NFS200.IFS
DEVICE=C:\OSTCP200\BIN\VDOSTCP.VDD
DEVICE=C:\OSTCP200\BIN\VDOSTCP.SYS
RUN=C:\OSTCP200\BIN\VDOSTCTL.EXE
REM *** TCP/IP SET COMMANDS
DEVICE=C:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=C:\IBMCOM\PROOCOL\LANDLLDD.OS2
RUN=C:\IBMCOM\PROTOCOL\LANDLL.EXE
DEVICE=C:\OS2\LOG.SYS
RUN C:\OS2\EPWROUT.EXE 1
RUN C::\OS2\EPW.EXE
DEVICE=C:\CMLIB\ACSLANDD.SYS
DEVICE\C:\CMLIB\CMKFMDE.SYS
```



PROTOCOL.INI

```
[PROTMAN_MOD]
   DRIVERNAME = PROTMAN$

[IBMLXCFG]
   LANDD_nif = LANDD.NIF
   TCPIP_nif = TCPIP.NIF
   IBMTOK_nif = IBMTOK.NIF


[TCPIP_V21]
   DRIVERNAME = TCPIP$
   BINDINGS = IBMTOK_NIF

[LANDD_nif]
   DRIVERNAME =  LANDD$
   BINDINGS = BIMTOK_nif
   ETHERAND TYPE = "I"
   MAX_SAPS =  3

[IBMTOK_nif]
   DRIVERNAME = IBMTOK$
   ADAPTER = "PRIMARY"
   NETADDRESS = '400012345678"
```


© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

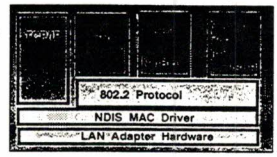
Connecting people to information through technology

Mixed Environment - LANSUP

TCP/IP, CM/2, NetWare, OS/2

CONFIG.SYS

```
DEVICE=C:\IBMCOM\PRTOOCOL\LANPDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
DEVICE=C:\C:\MMOS2\ADSHDD.SYS
DEVICE=C:\MMOS2\SMVDD.SYS
RUN=C:\IBMCOM\PRTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
DEVICE=C:\IBMCOM\PROTOCOL\INET.SYS
DEVICE=C:\IBMCOM\PROTOCOL\IFNDIS.SYS
DEVICE=C:\IBMCOM\MACS\IBMTOK.OS2
REM *** SERIES OF SET COMMANDS FOR TCP ****
RUN=C:\OSTCP200\BIN\CNTRL.EXE
IFS=C:\OSTCP200\BIN\NFS200.IFS
DEVICE=C:\OSTCP200\BIN\VDOSTCP.VDD
DEVICE=C:\OSTCP200\BIN\VDOSTCP.SYS
RUN=C:\OSTCP200\BIN\VDOSTCTL.EXE
REM *** TCP/IP SET COMMANDS
DEVICE=C:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=C:\IBMCOM\PROOCOL\LANDLLDD.OS2
RUN=C:\IBMCOM\PROTOCOL\LANDLL.EXE
DEVICE=C:\OS2\LOG.SYS
RUN C:\OS2\EPWROUT.EXE 1
RUN C::\OS2\EPW.EXE
DEVICE=C:\CMLIB\ACSLANDD.SYS
DEVICE\C:\CMLIB\CMKFMDE.SYS
```




```
REM --- NetWare Requester Statements BEGIN ----
SE NWLANGUAGE=ENGLISH
DEVICE=C:\NWARE\LSL\SYS
RUN=C:\NWARE\DDAEMON.EXE
DEVICE=C:\NWARE\LANSUP.SYS
DEVICE=C:\NWARE\ROUTE.SYS
DEVICE=C:\NWARE\IPX.SYS
DEVICE=C:\NWARE\SPX.SYS
RUN=C:\NWARE\SPDAEMON.EXE
DEVICE=C:\NWARE\NWREQ.SYS
IFS=C:\NWARE\NWIFS.SYS
RUN=C:\NWARE\NWDAEMON.EXE
DEVICE=C:\NWARE\VIPX.SYS
DEVICE=C:\NWARE\VSHELL.SYS PRIVATE
REM --- NetWare Requester Statements End ---
```

PROTOCOL.INI

```
[PROTMAN_MOD]
  DRIVERNAME = PROTMAN$
[IBMLXCFG]
  LANDD_nif = LANDD.NIF
  TCPIP_nif = TCPIP.NIF
  IBMTOK_nif = IBMTOK.NIF
[TCPIP_V21]
  DRIVERNAME = TCPIP$
  BINDINGS = IBMTOK_NIF
[LANDD_nif]
  DRIVERNAME = LANDD$
  BINDINGS = BIMTOK_nif
  ETHERAND TYPE = "I"
  MAX_SAPS = 3
[IBMTOK_nif]
  DRIVERNAME = IBMTOK$
  ADAPTER = "PRIMARY"
  NETADDRESS = "400012345678"
```

NET.CFG

```
Netware requester
  preferred server pita
Protocol IPX
  bind lansup
Link driver lansup
  frame token-ring
  frame token-ring_snap
  node address 400012345678
```

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT019B

IBM Networking

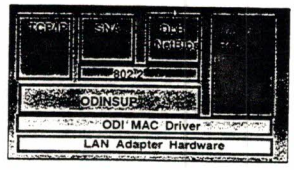
Connecting people to information through technology

Mixed Environment - ODINSUP

TCP/IP, CM/2, NetWare, OS/2

CONFIG.SYS

```
DEVICE=C:\IBMCOM\PRTOOCOL\LANPDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
REM --- NetWare Requester Statements BEGIN ----
SE NWLANGUAGE=ENGLISH
DEVICE=C:\NWARE\LSL\SYS
RUN=C:\NWARE\DDAEMON.EXE
DEVICE=C:\NWARE\token.SYS
DEVICE=C:\NWARE\ODINSUP.SYS
DEVICE=C:\NWARE\ROUTE.SYS
DEVICE=C:\NWARE\IPX.SYS
DEVICE=C:\NWARE\SPX.SYS
RUN=C:\NWARE\SPDAEMON.EXE
DEVICE=C:\NWARE\NWREQ.SYS
IFS=C:\NWARE\NWIFS.SYS
RUN=C:\NWARE\NWDAEMON.EXE
DEVICE=C:\NWARE\VIPX.SYS
DEVICE=C:\NWARE\VSHELL.SYS PRIVATE
REM --- NetWare Requester Statements End ---
DEVICE=C:\C:\MMOS2\ADSHDD.SYS
DEVICE=C:\MMOS2\SMVDD.SYS
RUN=C:\IBMCOM\PRTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
DEVICE=C:\IBMCOM\PROTOCOL\INET.SYS
DEVICE=C:\IBMCOM\PROTOCOL\IFNDIS.SYS
REM DEVICE=C:\IBMCOM\MACS\IBMTOK.OS2
```




```
REM *** SERIES OF SET COMMANDS FOR TCP ****
RUN=C:\OSTCP200\BIN\CNTRL.EXE
IFS=C:\OSTCP200\BIN\NFS200.IFS
DEVICE=C:\OSTCP200\BIN\VDOSTCP.VDD
DEVICE=C:\OSTCP200\BIN\VDOSTCP.SYS
RUN=C:\OSTCP200\BIN\VDOSTCTL.EXE
REM *** TCP/IP SET COMMANDS
DEVICE=C:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=C:\IBMCOM\PROOCOL\LANDLLDD.OS2
RUN=C:\IBMCOM\PROTOCOL\LANDLL.EXE
DEVICE=C:\OS2\LOG.SYS
RUN C:\OS2\EPWROUT.EXE 1
RUN C::\OS2\EPW.EXE
DEVICE=C:\CMLIB\ACSLANDD.SYS
DEVICE\C:\CMLIB\CMKFMDE.SYS
```

PROTOCOL.INI

```
[PROTMAN_MOD]
  DRIVERNAME = PROTMAN$
[IBMLXCFG]
  LANDD_nif = LANDD.NIF
  TCPIP_nif = TCPIP.NIF
[TCPIP_V21]
  DRIVERNAME = TCPIP$
  BINDINGS = TOKEN
[LANDD_nif]
  DRIVERNAME = LANDD$
  BINDINGS =TOKEN
  ETHERAND TYPE = "I"
  MAX_SAPS = 3
[TOKEN]
  DRIVERNAME = TOKEN$
  ADAPTER = "PRIMARY"
  NETADDRESS = "400012345678"
```

NET.CFG

```
Netware requester
  preferred server pita
Protocol ODINSUP
  bind TOKEN
Link driver token
  frame token-ring
  frame token-ring_snap
  node address 400012345678
```

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT019C

Mixed Environment - ODI2NDI

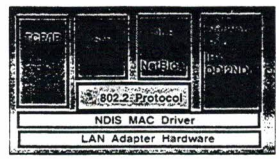
IBM Networking

Connecting people to information through technology

TCP/IP, CM/2, NetWare, OS/2

CONFIG.SYS

```
DEVICE=C:\IBMCOM\PRTOCOL\LANPDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
REM *** SERIES OF SET COMMANDS FOR TCP ****
RUN=C:\OSTCP200\BIN\CNTRL.EXE
IFS=C:\OSTCP200\BIN\NFS200.IFS
DEVICE=C:\OSTCP200\BIN\VDOSTCP.VDD
DEVICE=C:\OSTCP200\BIN\VDOSTCP.SYS
RUN=C:\OSTCP200\BIN\VDOSTCTL.EXE
REM *** TCP/IP SET COMMANDS
DEVICE=C:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=C:\IBMCOM\PROOCOL\LANDLDD.OS2
RUN=C:\IBMCOM\PROTOCOL\LANDLL.EXE
DEVICE=C:\OS2\LOG.SYS
RUN C:\OS2\EPWROUT.EXE 1
RUN C::\OS2\EPW.EXE
DEVICE=C:\CMLIB\ACSLANDD.SYS
DEVICE\C:\CMLIB\CMKFMDE.SYS
DEVICE=C:\C:\MMOS2\ADSHDD.SYS
DEVICE=C:\MMOS2\SMVDD.SYS
RUN=C:\IBMCOM\PRTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
DEVICE=C:\IBMCOM\PROTOCOL\NET.SYS
DEVICE=C:\IBMCOM\PROTOCOL\IFNDIS.SYS
DEVICE=C:\IBMCOM\MACS\IBMTOK.OS2
```




```
REM --- NetWare Requester Statements BEGIN ----
SE NWLANGUAGE=ENGLISH
DEVICE=C:\NWARE\LSL\SYS
RUN=C:\NWARE\DDAEMON.EXE
DEVICE=C:\NWARE\ROUTE.SYS
DEVICE=C:\NWARE\IPX.SYS
DEVICE=C:\NWARE\SPX.SYS
DEVICE=C:\NWARE\ODI2NDI
RUN=C:\NWARE\SPDAEMON.EXE
DEVICE=C:\NWARE\NWREQ.SYS
IFS=C:\NWARE\NWIFS.SYS
RUN=C:\NWARE\NWDAEMON.EXE
DEVICE=C:\NWARE\VIPX.SYS
DEVICE=C:\NWARE\VSHELL.SYS PRIVATE
REM --- NetWare Requester Statements End ---
```

PROTOCOL.INI

```
[PROTMAN_MOD]
   DRIVERNAME = PROTMAN$
[IBMLXCFG]
   LANDD_nif = LANDD.NIF
   TCPIP_nif = TCPIPNIF
   ODI2NDI_nif = ODI2NDI
   IBMTOK_nif = IBMTOK$
[TCPIP_V21]
   DRIVERNAME = TCPIP$
   BINDINGS = IBMTOK$
[LANDD_nif]
   DRIVERNAME =  LANDD$
   BINDINGS =IBMTOK$
   ETHERAND TYPE = "I"
   MAX_SAPS = 3
[IBMTOK_NIF]
   DRIVERNAME = IBMTOK$
   ADAPTER = "PRIMARY"
   NETADDRESS = "400012345678"
[ODI2NDI_NIF]
   DRIVERNAME = ODI2NDI$
   BINDINGS = IBMTOK$
   NETADDRESS = "400012345678"
   TOKEN-RING = "YES"
   TOKEN-RING_SNAP = "YES"
```

NET.CFG

```
Netware requester
  preferred server pita
```


© Copyright IBM Corporation 1993, 1994, 1995

IBM's Token-Ring Adapter Family

IBM Networking

Connecting people to information through technology

	ISA	MC	EISA	PCMCIA	PCI
Token-Ring	New	New	✔	New	New
Ethernet IEEE 802.3	✔	✔	New	✔	
FDDI	✔	✔	✔		
Wireless	New	New		New	
ATM	New	New			

ISA Adapters
IBM Token-Ring Network PC Adapter II
IBM Token-Ring 16/4 Adapter
IBM Token-Ring 16/4 Adapter II
New IBM Auto 16/4 Token-Ring ISA Adapter

MC Adapters
Auto LANStreamer MC 32 Adapter
LANStreamer MC 16 Adapter
New IBM Auto 16/4 Token-Ring MC Adapter

EISA Adapter
IBM Token-Ring Network 16/4 EISA Adapter

PCMCIA Adapters
New IBM Token-Ring Auto 16/4 Credit Card Adapter

PCI Adapter
New IBM Auto LANStreamer PCI Adapter

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

Token-Ring Adapter Positioning

	ISA	MC	EISA	PCMCIA
Servers (NDIS or ODI)	IBM Token-Ring 16/4 Adapter II	IBM Auto LANStreamer MC 32	IBM 16/4 EISA Busmaster	*IBM Token-Ring 16/4 Credit Card Adapter **
Bridge	IBM Auto 16/4 Token-Ring ISA	IBM Auto LANStreamer MC 32 or IBM LANStreamer MC 16		
OS/2	IBM Auto 16/4 Token-Ring ISA	IBM Auto LANStreamer MC 32 or IBM LANStreamer MC 16	IBM 16/4 EISA Busmaster	*Token-Ring 16/4 Credit Card Adapter **
DOS or Windows (DOS apps)	IBM Auto 16/4 Token-Ring ISA	Auto 16/4 Token-Ring MC Adapter	IBM 16/4 EISA Busmaster	*Token-Ring 16/4 Credit Card Adapter **
Windows (Win apps)	IBM Auto 16/4 Token-Ring ISA	IBM Auto LANStreamer MC 32 or IBM LANStreamer MC 16	IBM 16/4 EISA Busmaster	*Token-Ring 16/4 Credit Card Adapter **

© Copyright IBM Corporation 1993, 1994, 1995

IBM Networking

Connecting people to information through technology

IBM's Ethernet Adapter Family

MC Adapters
IBM LAN Adapter/A for Ethernet
IBM EtherStreamer MC 32 Adapter

ISA Adapters
IBM LAN Adapter for Ethernet
IBM LAN Adapter for Ethernet TP
IBM LAN Adapter for Ethernet CX

	ISA	MC	EISA	PCMCIA	PCI
Token-Ring	New	New	✔	New	New
Ethernet IEEE 802.3	✔	✔	New	✔	
FDDI	✔	✔	✔		
Wireless	New	New		New	
ATM	New	New			

PCMCIA Adapters
IBM Ethernet Credit Card Adapter II 10BaseT
IBM Ethernet Credit Card Adapter II 10Base2

EISA Adapters
New IBM Ethernet EISA Adapter

© Copyright IBM Corporation 1993, 1994, 1995

IBM's Wireless Adapter Family

IBM Networking

Connecting people to information through technology

	ISA	MC	EISA	PCMCIA	PCI
Token-Ring	New	New	✔	New	New
Ethernet IEEE 802.3	✔	✔	New	✔	
FDDI	✔	✔	✔		
Wireless	New	New		New	
ATM	New	New			

MC Adapters
New IBM Wireless LAN ISA/MCA Adapter

ISA Adapters
New IBM Wireless LAN ISA/MC Adapter

PCMCIA Adapters
New IBM Wireless LAN PCMCIA Adapter

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT030D

IBM's FDDI Adapter Family

IBM Networking

Connecting people to information through technology

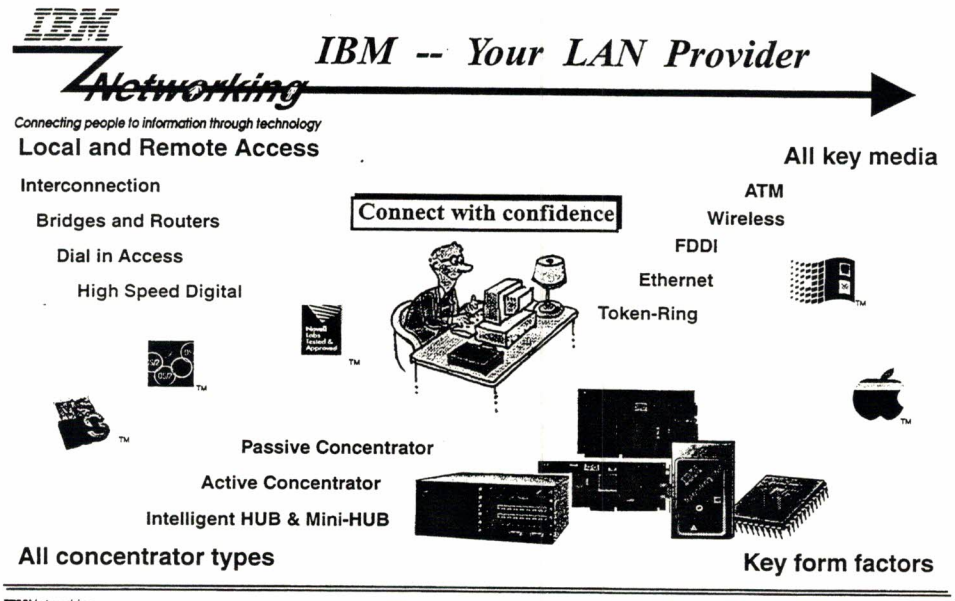
	ISA	MC	EISA	PCMCIA	PCI
Token-Ring	New	New	✔	New	New
Ethernet IEEE 802.3	✔	✔	New	✔	
FDDI	✔	✔	✔		
Wireless	New	New		New	
ATM	New	New			

EISA Adapters
IBM FDDI Fiber Base EISA Adapter
IBM FDDI Copper Base EISA Adapter
IBM FDDI Fiber Extender EISA/ISA Adapter
IBM FDDI Copper Extender EISA/ISA Adapter
New IBM FDDI UTP5 Base EISA Adapter

MC Adapters
IBM FDDI Fiber Base MC Adapter
IBM FDDI Copper Base MC Adapter
IBM FDDI Fiber Extender MC Adapter
IBM FDDI Copper Extender MC Adapter
New IBM FDDI UTP5 Base MC Adapter

ISA Adapters
IBM FDDI Fiber Base ISA Adapter
IBM FDDI Copper Base ISA Adapter
IBM FDDI Fiber Extender EISA/ISA Adapter
IBM FDDI Copper Extender EISA/ISA Adapter
New IBM FDDI UTP5 Base ISA Adapter

IBM Networking © Copyright IBM Corporation 1993, 1994, 1995 ADPT030E


IBM Networking
IBM -- Your LAN Provider
Connecting people to information through technology
Local and Remote Access
Interconnection
Bridges and Routers
Dial in Access
High Speed Digital
Connect with confidence
All key media
ATM
Wireless
FDDI
Ethernet
Token-Ring
Passive Concentrator
Active Concentrator
Intelligent HUB & Mini-HUB
All concentrator types
Key form factors
IBMNetworking
© Copyright IBM Corporation 1993, 1994
ADPT031A

MGT 0911
How to Introduce Objects to Your Organization

David Moskowitz
Productivity Solutions
Norristown, Penn.

How to Introduce Objects to your Organization

David Moskowitz
Productivity Solutions

Goals and Agenda

- To object or not to object...
- Understand what OOT means
- Know the steps to develop a plan
- Set a goal to learn from the first effort

Expectations

- If you have a question, ask it!
- Overview of a long term process
- There is no magic cookie

Why Objects?

- Reusable software
- Why not?
- Gui
- Management wants to
- Technical staff wants to
- Competition gives us no choice

You need a reason to change

Arguments against objects

- It's too hard to learn
- Others have had trouble
- It costs too much
- Techniques unproven
- The fad will pass

You can succeed...
...if you want to & you have a plan

The Myths of OOP

- Objects mean reuse
- Objects mean good user interface
- Objects are a magic cookie
- OOP is learning the language
- Every object is reusable
- OOP is a new way of doing what we do
- OOP is a fancy name for what we've done
- Any good professional can pick up OOT in 6 months

The plan -- 6 basic steps

- Pick a problem
- Make it short
- Do something measurable
- Pick a goal people can commit to
- Do it with available resources & budget
- Evaluate the results

Pick the right problem

- Something with a degree of urgency
- Not risky, but necessary
- Vital the team see it through to completion
- Need only a small team
- Develop experience

Problems to avoid

- Critical project, tight budget
- Too many roadblocks
- Long term view missing
- Support lacking
- No way to evaluate results

Quick achievement (short)

- Hold the team together
- Learning is easier with short duration
  - At most 15 months
- Low frustration level with initial problems
- Add more people sooner

Impact (measurable)

- Make a contribution
- Easier to evaluate results
- Project planned for success
- Get commitment
  - Staff commitment is more likely if the people believe in the results.

Available resources

- More likely to accomplish the task
- Attract less negative attention
- Consider outside mentoring
- Develop internal mentors
- Scale project size to resources
  - Number of people determine the number of likely classes (and vice versa)
  - 30 classes will take 1 person 1 to 3 months
  - 100 clases 2 to 3 people, 4 to 8 months
  - 100 => 5 to 12 people, 6 to 15 months

Evaluate results

- First effort not perfect
  - Don't plan to keep it forever
- Learn from mistakes
  - What worked, what didn't
- Iterative development
- Two parallel efforts & compare
  - OOP
  - Traditional
  - When you finish, make changes

Tradeoffs I

- Low risk & reward
  - Self-help
  - Self-study
  - 1 person project
  - Some experience
- Low risk, moderate reward
  - Moderate cost
  - Low impact project
  - Small team
  - OOD & OOP
  - Mentoring

Tradeoffs II

- Moderate risk & reward
  - Medium size project
  - Re-engineer
  - Training
  - Mentoring
- Moderate risk, high reward
  - Strategic project
  - Training & mentoring critical
  - Hire experts

Basic goals

- Effective use of OOT
- Design and build many classes
- Learn the tools
- Learn the class libraries
- No longer fight OOT
- Prefer reuse versus build from scratch

Keys to success

- Process not a quick fix
- Start with reasonable goals
- Control expectations
- Personal mastery & development
- Keep clients & management informed
- Control project growth
- Have a plan, before you start

The other stuff

- Evaluate possibilities
- Study class libraries
  - –You can't reuse what you can't find
- Record ideas accepted & rejected
  - –Today's reject may be a solution later
- There is no such thing as a perfect class -- the process must expect (almost encourage) failure enroute to successful refinement of the architecture.

Evaluate possibilities

- Pick a language and tools
  - –If you have COBOL programmers, C++ might not be the best place to start.
- Think about ramifications on people
- Involve end-users in the design.
- It helps to have a strong architectural vision.

Demonstrations

- Celebration of achievement
- Helps develop focus
- Use it to re-architect & re-design
- Do not treat it as a drop-dead point

Method mentor

- Facilitator
- Be familiar with more than one
  - Booch-Rumbaugh, Coad, CRC, etc.
- Method support tools
- Work with small team
- Small team work with others

The role of mistakes

Mistakes are vital but only if you learn from them.

Best if your own team can spot its own mistakes.

If you find the application getting too complex -- add classes.

Questions

- I've never heard a dumb question!
- The only dumb question is the one that isn't asked!
- Something to consider:

The change associated with OOT means moving away from mapping the problem as a series of functions to perceiving the problem as patterns of collaborating objects with cooperating behaviors & responsibilities.

Thank you

For more information:

David Moskowitz
Productivity Solutions, Inc.
164 Avondale Road
Norristown, PA 19403-2952
Voice: 610-631-5685
CompuServe: 76701,100
Internet: dmoskowitz@cis.CompuServe.com
Internet: davidm2@ibm.net

this page intentionally left blank

PRO 0905
The Power of Frameworks

David Sharp
IBM

INTRODUCING OBJECT-ORIENTED FRAMEWORKS

Through its use of frameworks, Taligent, Inc., an independent joint venture of Apple Computer, Inc., IBM Corporation, and Hewlett-Packard Company, is realizing the full promise of object technology. A framework defines the behavior of a collection of objects, providing an innovative way to reuse both software designs and code.

> A framework is a set of prefabricated software building blocks that programmers can use, extend, or customize for specific computing solutions. Taligent has designed frameworks for system software functions, such as networking, multimedia and database access. With frameworks, software developers don't have to start from scratch each time they write an application. Frameworks are built from a collection of objects, so both the design and code of a framework may be reused.
> - Taligent Glossary

Presenting a process for developing frameworks, this paper highlights four important guidelines:

- Derive frameworks from existing problems and solutions
- Develop small, focused frameworks
- Build frameworks using an iterative process driven by client participation and prototyping
- Treat frameworks as products by providing documentation and support, and by planning for distribution and maintenance

The information is organized into three sections. "Understanding Frameworks" describes different types of frameworks and how they are used. "Maximizing Framework Benefits" addresses organizational concerns that impact framework development. "Developing Frameworks" outlines a process for identifying and designing frameworks. Each section also includes examples of how frameworks are currently being developed and deployed. At the end of this paper, you'll find a brief summary, some tips for starting to build your own frameworks, and a reading list.

While the process and techniques discussed here are ideal for developing in the Taligent™ Application Environment, you'll find that they can also be applied to other object-oriented programming projects.

Building Object-Oriented Frameworks is intended primarily for software developers and designers in commercial, corporate, and higher education software development organizations. However, hardware designers, strategic technology planners, and technical managers might also find it useful. Two earlier Taligent white papers, The Intelligent Use of Objects and Leverging Object-Oriented Frameworks, provide an introduction to object-oriented design and frameworks for those who are new to the technology.

UNDERSTANDING FRAMEWORKS

A framework captures the programming expertise necessary to solve a particular class of problems. Programmers purchase or reuse frameworks to obtain such problem-solving expertise without having to develop it independently.

> A framework helps developers provide solutions for problem domains and better maintain those solutions. It provides a well-designed and thought out infrastructure so that when new pieces are created, they can be substituted with minimal impact on the other pieces in the framework.- Nelson, IEEE Computer, 1994

Developing a framework differs from developing a standalone application. A successful framework solves problems that, on the surface, are quite different from the problem that justified its creation.

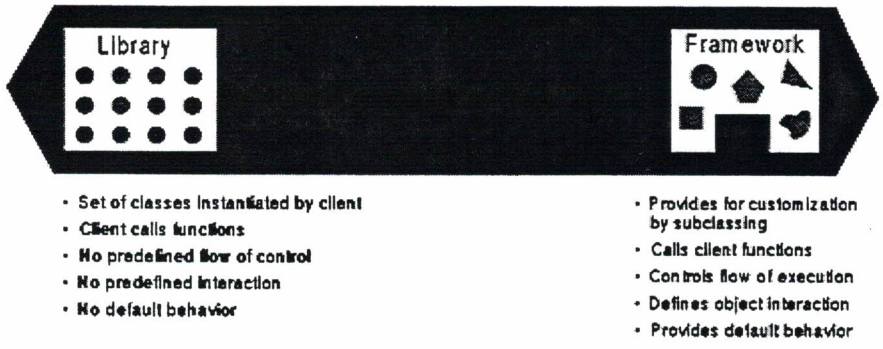
The problem-solving expertise must be captured so that it is independent of both the original problem and the future solutions in which it is used; however, each program that uses the framework should appear to be the one for which it was designed.

You have to clearly identify the class of problem a framework addresses. For your clients to adapt the framework to new problems, they must understand the solution the framework provides and how to incorporate it into their programs. Because others have to understand how to use your frameworks, it is critical that you follow good software design practices.

Describing Frameworks

While clear differences exist between class libraries and frameworks, some libraries exhibit framework-like behavior, and some frameworks can be used like class libraries. You can view this as a continuum, with traditional class libraries at one end and sophisticated frameworks at the other:

LIBRARY-FRAMEWORK CONTINUUM



Frameworks can be characterized by the problem domains that they address, as well as by their internal structure.

FRAMEWORK DOMAINS

The problem domain that a framework addresses can encompass application functions, domain functions, or support functions:

- **Application frameworks** encapsulate expertise applicable to a wide variety of programs. These frameworks encompass a horizontal slice of functionality that can be applied across client domains. Current commercial graphical user interface (GUI) application frameworks, which support the standard functionality required by all GUI applications, are one type of application framework. (The Apple MacApp[a] system and Borland's OWL[a] system are two such frameworks.)
- **Domain frameworks** encapsulate expertise in a particular problem domain. These frameworks encompass a vertical slice of functionality for a particular client domain. Examples of domain frameworks include: a control systems framework for developing control applications for manu-facturing, securities trading framework, multimedia framework, or data access framework.

- **Support frameworks** provide system-level services, such as file access, distributed computing support, or device drivers. Application developers typically use support frameworks directly or use modifications produced by systems providers. However, even support frameworks can be customized--for example when developing a new file system or device driver.

The Taligent Application Environment frameworks extend across all three categories. These frameworks reduce the amount of code developers have to write and maintain.

Similarly, you can create frameworks that capture your own problem-domain expertise. Whether you are developing custom applications in a corporate setting or developing a suite of commercial applications as an independent software vendor, building and using frameworks can increase productivity. The frameworks you build will usually be domain or application frameworks.

Framework Structures

Identifying the high-level structure of a framework can make it easier to describe the behavior of the framework and provides a starting point for designing framework interactions.

For example, some frameworks can be described as manager-driven--a single controlling function triggers most of the framework actions. You call the controlling function to start the framework and the framework creates the necessary objects and calls the appropriate functions to perform a specific task.

An application framework, for example, generally uses a manager object to take input events from the user and distribute them to the other objects in the framework.

Another categorization is based on how a framework is used--whether you derive new classes or instantiate and combine existing classes. This distinction is sometimes referred to as architecture-driven versus data-driven, or inheritance-focused versus composition-focused.

Using and Extending a Framework

Using the Framework

```
* Set box color
* Give box contents
* Draw box and contents

Drawing a default check box

TBox box;
box.SetColor(TRGBColor(0,1,0);
TCheckContents check;
box.SetContents(check);
box.Draw();
```

Extending the Framework

```
* Derive from contents class
  and override DrawContents

Extending the framework to draw
an X instead of the default check
class TXMarkContent : public TboxContent{
public:
  virtual void DrawContents(TGrafPort&port)
};
void
TxMarkContent::DrawContents(TGrafPort&port
{
   TLine line1(TGPoint(0,0), TGPoint(10,10)
   TLine line2(TGPoint(10,0), TGPoint(0,10)
 line1.Draw(port);
   line2.Draw(port);
}
```

Architecture-driven frameworks rely on inheritance for customization. Clients customize the behavior of the framework by deriving new classes from the framework and overriding member functions.

Data-driven frameworks rely primarily on object composition for customization. Clients customize the behavior of the framework by using different combinations of objects. The objects that clients pass into the framework affect what the framework does, but the framework defines how the objects can be combined.

Frameworks that are heavily architecture-driven can be difficult to use because they require clients to write a substantial amount of code to produce interesting behavior. Purely data-driven frameworks are generally easy to use, but they can be limiting.

One approach for building frameworks that are both easy to use and extensible is to provide an architecture-driven base with a data-driven layer. Most frameworks provide ways for clients to both use the built-in functionality and modify or extend that functionality.

Typically, clients use a framework's built-in functionality by instantiating classes and calling their member functions. Clients extend and modify a framework's functionality by deriving new classes and overriding member functions.

Take a simple framework that supports, among other things, drawing shapes. Clients might use it to draw a check box or extend it to draw other types of boxes. (See "Using and Extending a Framework" below.)

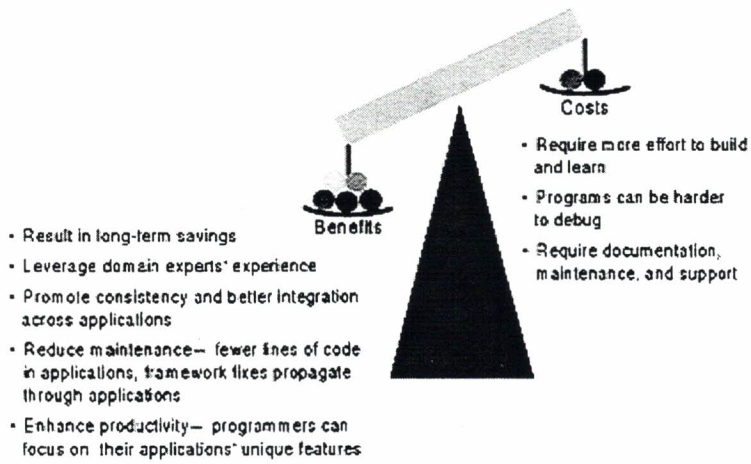
MAXIMIZING FRAMEWORK BENEFITS

The key to maximizing the benefits of writing frameworks is to get as many developers as possible using them. Frameworks are a long-term investment; the benefits gained from developing frameworks are not necessarily immediate:

- Framework designers need more time to create a framework than a procedural library.
- Clients need more time to learn a framework than a procedural library.

For framework development to be successful, it must be supported by your team's processes and organization.

Framework Benefits



> The most profoundly elegant framework will never be reused unless the cost of understanding it and then using its abstractions is lower than the programmer's perceived cost of writing them from scratch.
> -Booch, Dr. Dobb's Journal, 1994

Managing Dependencies

Projects can often be factored into a number of separate frameworks and assigned to small teams. If it takes more than three or four programmers to produce a framework, it can probably be split into a set of smaller frameworks. Teams of two to four are usually more effective than teams of one, unless the single programmer is both an experienced framework developer and a domain expert.

Working with several small teams introduces additional challenges:

- Programmers are focused on one aspect of a large project and might not understand all of the interrelationships and client implications.
- Architectural consistency must be maintained across teams.
- Dependencies between frameworks can create bottlenecks.

To alleviate these problems, you can:

- Appoint a project architect who maintains the "big picture" and ensures that the frameworks ultimately work together
- Follow standard design and coding guidelines
- Decouple the frameworks by isolating the dependencies in intermediary classes

Often when one framework requires the services of another framework, the connection can be implemented through an interface or server object. Then, only one object is dependent on the other framework. Until the other framework can support the necessary operations, the intermediary class provides stub code that allows the rest of the framework to be tested. Loosely-coupled frameworks are generally more flexible from the client's perspective as well.

This same approach can be used to isolate platform-dependent code, or code that accesses a particular applications framework. To use different platforms or application frameworks, only one piece of the framework needs to be changed. Intermediaries can also be used to access legacy data or nonframework services.

Delivering a Framework as a Product

Even if your framework will only be used internally, you must treat it like a product. Documenting your framework is an important step in the development cycle, but you must also plan how the finished product will be distributed and supported.

Distributing and Promoting Frameworks

To use your frameworks, other developers need to know that they exist and know how to access them. Unless a process is established for providing and distributing frameworks, it is difficult to get developers to use them. Reuse doesn't just happen, your organization has to actively support and encourage it. One way to do this is to adjust the reward structures - reward programmers for writing and distributing frameworks that others can use. Even more important, reward the programmers who use them.

Taligent Standards
Early on, Taligent realized the need for common coding standards throughout the company. The chief architect began compiling a list of do's and don'ts that were required reading for all new engineers. This compilation has evolved into Taligent`s Guide to Designing Programs: Well-Mannered Object-Oriented Design in C++, published by Addison-Wesley. This book, which contains the guidelines followed by the Taligent engineers in the development of the Taligent Application Environment, provides an excellent basis for your own coding standards.

Ideally, all frameworks are kept in a central repository and a repository manager is responsible for notifying clients about new frameworks and updates to old ones. With a large enough repository, selecting the appropriate frameworks becomes an integral part of developing new program solutions.

Supporting Frameworks

To realize the benefits frameworks can provide, your organization has to commit resources to support them. You must be able to assist your clients and respond to their problems and requests.

Over the lifetime of a framework, the cost of supporting it actually becomes a benefit--the support cost of one framework with three dependent applications will be less than the cost of supporting three independent applications with duplicate code. The more applications that use a framework, the bigger the savings. Long term, using frameworks can reduce support and maintenance costs.

Early in its life, a framework will probably require routine maintenance to fix bugs and respond to client requests. Over time, even the best framework will probably need to be updated to support changing requirements.

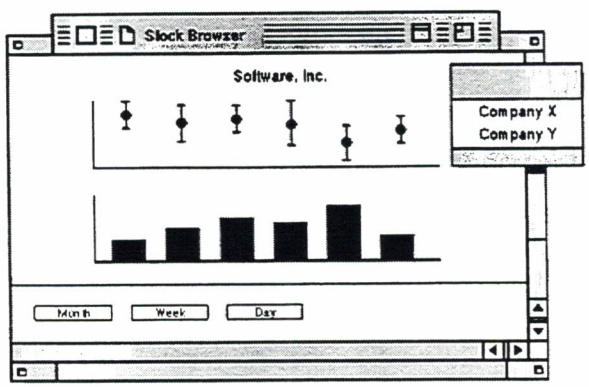
DEVELOPING FRAMEWORKS

The first step in developing a framework is to analyze your problem domain and identify the frameworks you need. For each framework, you:

- Identify the primary abstractions
- Design how clients interact with the framework
- Implement, test, and refine the design

To illustrate these steps, this section describes a small program and one of the frameworks that could be used to build it. The program is a data-visualization tool called StockBrowser that allows users to browse stock histories and display the information graphically. This program, allows users to select different stocks and view the price trading volume graphically. The stock information can be displayed as daily, weekly, or monthly values, showing users information such as how a stock's price ranged on a particular day or what month had the heaviest volume of trading:

Stock Browser



Once you've identified the problem domain, break the problem down into a collection of frameworks that can be used to build a solution.

If the problem maps to a process, describe the process from the user's perspective. For each framework, identify the process it models. Once you've outlined the process, you should be able to identify the necessary abstractions.

IDENTIFYING FRAMEWORKS

Frameworks can be thought of as abstractions of possible solutions to problems. You can often identify opportunities for new frameworks by analyzing existing solutions.

To determine what frameworks you need, think in terms of families of applications rather than individual programs:

- Look for software solutions that you build repeatedly, particularly in key business areas
- Identify what the solutions have in common and what is unique to each program

The common pieces--the parts that are constant across programs--become the foundation for your frameworks. Factor these pieces into small, focused frameworks

.

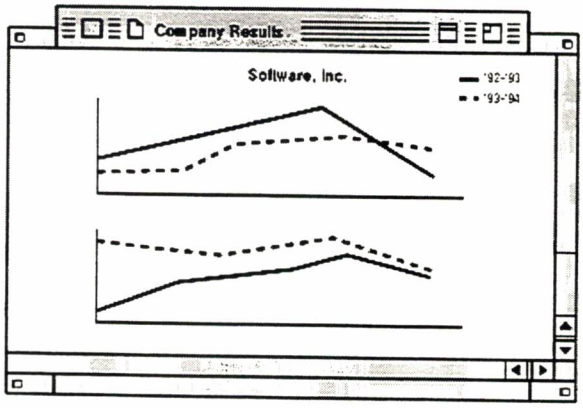
Where there's a suite of applications that solve similar problems, there's an opportunity for developing a framework. Look for potential frameworks in:

- Real-world models
- Processes performed by end users
- Source code for current software solutions

One of the frameworks that could be used to build this program is a 2-D graphing framework. The graphing framework would provide support for drawing and labeling the axes and for plotting the data.

Such a framework could be used in a wide variety of data-visualization applications--from executive information systems to scientific visualization programs. For example, you might need to build another program that charts sales volume information:

Company Results



Identify the abstractions your clients need to describe their problems and then provide the logic for producing a valid solution with those abstractions.

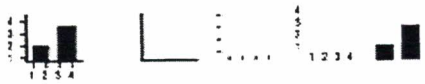
The easiest way to identify the abstractions is with a bottom-up approach--start by examining existing solutions. Analyze the data structures and algorithms and then organize the abstractions. Always identify the objects before you map out the class hierarchy and dependencies.

If you are familiar with the problem domain, you can draw from your past experience and former designs to identify abstractions and begin designing your framework.

If you're not an expert in the problem domain, or haven't developed any applications for it, examine applications written by others and consider writing an application in the domain. Then factor out the common pieces and identify the abstractions.

For the graphing framework, there are two primary abstractions: the graph itself and drawers for the graph. Separating the drawing information from the graph is what makes it possible to extend this simple framework. New graph elements can be added by providing new types of drawers.

Graphing Framework Abstractions

GRAPH	DRAWERS
	
- Standard Graphs	- Axis Drawer - Mark Drawer - Label Drawer - Bar Drawer
Embodies the graph as a whole; provides the point of control.	Embodies individual graph elements; controls drawing and characteristics of each element.

> Good designers know many design patterns and techniques that they know lead to good designs. Applying recurring patterns to the design of a framework is one form of reuse. Using formalized "design patterns" also helps to document the framework, making it easier for clients to understand, use, and extend the framework.-Johnson, OOPSLA `93 Conference Proceedings, 1993

Design patterns identify, name, and abstract common themes in object-oriented design. They capture the intent behind a design by identifying objects, how objects interact, and how responsibilities are distributed among them. They constitute a base of experience for building reusable software, and they act as building blocks from which more complex designs can be built.

Architect Christopher Alexander first introduced the concept of patterns as a tool to encode the knowledge of the design and construction of communities and buildings. AlexanderÕs patterns describe recurring elements and rules for how and when to create the patterns. Designers of object-oriented software have begun to embrace this concept of patterns and use it as a language for planning, discussing, and documenting designs.

Just as you perform object decomposition to get from a high level design to an object-oriented implementation, you can decompose a framework into recurring patterns. Some patterns are generic, and some are specific to a problem domain. Each pattern can be characterized by its:

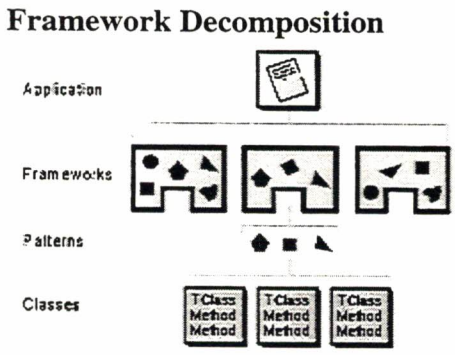
- Preconditions - the patterns that must be satisfied for this pattern to be valid.
- Problem - the problem addressed by the pattern.
- Constraints - the conflicting forces acting on any solution to the problem and the priorities of those constraints.
- Solution - the solution to the problem.

By describing the design of a framework in terms of patterns, you describe both the design and the rationale behind the design.

Reusing common patterns opens up an additional level of design reuse, where the implementations vary, but the micro-architectures represented by the patterns still apply

- Gamma, Helm, Johnson, and Vlissades, European Conference on Object-Oriented Programming, 1993

Framework Decomposition



DESIGNING THE FRAMEWORK

To be successful, you should design your frameworks to be:

- **Complete** - frameworks support features needed by clients and provide default implementations and built-in functionality where possible. Provide concrete derivations for the abstract classes in your frameworks and default member function implementations to make it easier for your clients to understand the framework and allow them to focus on the areas that they need to customize.
- **Flexible** - abstractions can be used in different contexts.
- **Extensible** - clients can easily add and modify functionality. Provide hooks so your clients can customize the behavior of the framework by deriving new classes.
- **Understandable** - client interactions with the frameworks are clear, and the frameworks are well-documented. Follow standard design and coding guidelines and provide sample applications that demonstrate the use of each framework. If the developers who build frameworks and those who use them follow the same guidelines, it facilitates reuse in both directions.

The most important consideration when you're designing frameworks is ease of use for the client programmer. From the client's perspective, an easy-to-use framework performs useful functions with no effort. The framework

works with little or no client code, even if the default implementations are simply placeholders, and it supports small, incremental steps to get from the default behavior to sophisticated solutions. However, it's harder for clients to work around bad abstractions than invent their own. If you don't know how to solve a problem in a reusable way inside your framework, leave it out.
A framework doesn't necessarily have to meet all of these requirements to be successful, but if it does, it will be easier to convince other programmers to use the framework.

When you design a framework, you should also look for ways to minimize the potential for client errors and enhance portability:

- Simplify clients' interactions with the framework to help prevent client errors. Make it as clear as possible in both your interfaces and documentation what's required of your clients.
- Isolate platform-dependent code to make it easier to port your framework. Designing for portability reduces the impact porting has on your clients.

If the framework models a process, determine which steps in the process the framework performs, and which steps the client performs.

As you begin to design how the framework will work, you might also discover that you can break the framework down into a collection of recurring design patterns, much the way you decompose the initial problem into a set of frameworks.

Each design pattern is a micro-architecture for a recurring element. Patterns can represent generic software elements, or elements that are particular to a problem domain. When you design a framework, look for recurring patterns that can be applied to other problems.

Designing the Client-Framework Interactions

In your framework design, focus on how the client interacts with the framework--what classes and member functions does the client use? Look for ways to reduce the amount of code that clients must write:

- Provide concrete implementations that can be used directly
- Minimize the number of classes that must be derived
- Minimize the number of member functions that must be overridden

One of the things to consider is how to support customization. Customization is a two-edged sword--you want to provide as flexible a framework as possible, but you also want to maintain the focus of the framework and minimize the complexity for the client. If you provide a completely flexible framework, it will be difficult for your clients to learn, and difficult for you to support.
One approach is to build a very flexible, general framework from which you derive additional frameworks for narrower problem domains. These additional frameworks provide the default behavior and built-in functionality, while the general framework provides the flexibility.

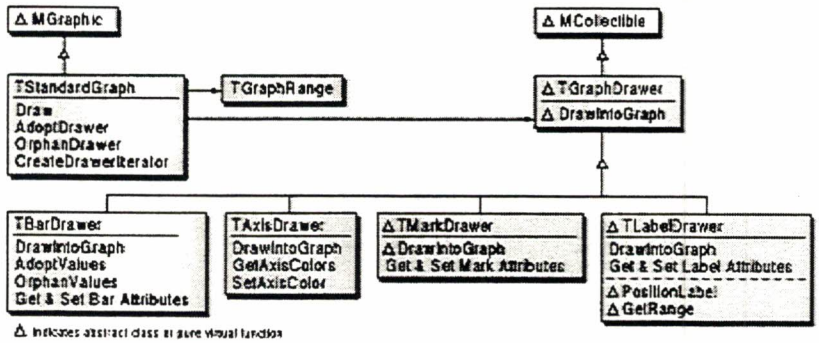
As you design the framework, also consider how the design can help communicate how to use the framework. Class names, functions names, and how you use pure virtual functions can all provide cues for using the framework.

You also need to determine how the framework classes and member functions interact with client code:

- What objects are created when the client calls framework functions?
- When are client overrides called by the framework?

As with most frameworks, the graphing framework provides both features that the client can use directly and an underlying structure that can be extended by deriving new classes--in other words, even this simple framework has both data-driven characteristics and architecture-driven characteristics.

GRAPHING FRAMEWORK ARCHITECTURE

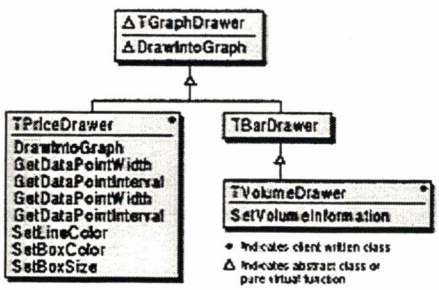


To use the framework, a client constructs the necessary drawers and a standard graph. When the client calls TStandardGraph::Draw, DrawIntoGraph is called for each drawer and the graph and all its elements are drawn. To change the appearance of the graph, clients can call functions such as TAxisDrawer::SetAxisColor.

To extend the framework, clients derive new drawers from TGraphDrawer or one of its subclasses. It's also possible to derive new graph types from TStandardGraph -for example, a new type of graph that supports buffered drawing.

For the StockBrowser program, two new drawers were added: TPriceDrawer and TVolumeDrawer. (See "StockBrowser Extensions" below.)

STOCK BROWSER EXTENSIONS



TPriceDrawer charts stock prices, displaying high, low, and close prices for a particular stock. This class is derived directly from TGraphDrawer and from MStockHolder, which holds a collection of daily stock data. TPriceDrawer overrides TGraphDrawer::DrawIntoGraph, and provides functions for manipulating the data points and attributes.

TVolumeDrawer charts stock trading volumes in a bar chart format. This class is derived from TBarDrawer, a generic 2-D bar chart element, and MStockHolder. TVolumeDrawer adds the function SetVolumeInformation to get the stock volume information to display.

To display the stock data, StockBrowser creates drawers for the graph elements, specifies attributes for the elements, and calls TStandardGraph::Draw. The graphing framework does the rest. The following sample code demonstrates this point.

```
TGraphRange xAxisRange (0, 53);
TGraphRange yAxisRange (0, 100);
TGPoint axisLengths (530, 100);

TStandardGraph graph(xAxisRange,
        yAxisRange, axisLengths);
.
.
GraphValue dataPointWidth = 6;
GraphValue dataPointInterval = 1;

TPriceDrawer* priceDrawer = new TPriceDrawer(dataPointWidth,
        dataPointInterval, *data);
priceDrawer->SetLineColor(TRGBColor(.75, .2, .75));
priceDrawer->SetBoxColor(TRGBColor(0, 0, 0));
graph.AdoptDrawer(priceDrawer); . . graph.Draw(*port);
```

Experiences from the ET++ SwapsManager Project

One example of a domain specific framework is the ET++ SwapsManager developed by the Union Bank of Switzerland. This framework was developed as a calculation engine for financial trading software.

The ET++ SwapsManager provides a model for swaps trading that demonstrates the advantages of applying framework technology to financial engineering.

The developers first analyzed how financial instruments and valuation procedures could be abstracted and then identified their commonalities and interrelationships. Though they had in mind a general framework for financial engineering, for their initial project, they focused on a subset of their problem domain - swaps trading. Their goal was to develop a specific solution that they could expand on.

The developers believe that the development of a framework starts with the implementation of a specific solution in a particular domain. You then successively rework the solution to cover a family of applications.

Guidelines for generalizing a design:

- Consolidate similar functionality across the system and implement it through a common abstraction.
- Divide large abstractions into several smaller abstractions. Each of the smaller abstractions should have a small, focused set of responsibilities.
- Implement each variation of an abstraction as an object. This increases the flexibility of the design.
- Use composition instead of inheritance where possible. This reduces the number of classes and reduces the complexity for the framework client

- Birrer and Eggenschwiler, European Conference on Object-Oriented Programming, 1993

Refining the Frameworks

As your framework takes shape, continually look for ways to refine it by adding more default behavior, and additional ways for users to view and interact with the data.

Building a framework is an iterative process. Beginning with your initial design, you should work with your clients to determine how the framework can be improved--implement features, test them, and verify them with your clients.

During this process, you'll go back and reanalyze the problem domain and refine your design based on testing, client feedback, and your own insights.

As your framework matures, you'll probably find more features to add and identify oppor-tunities for additional frameworks. These might be entirely new frameworks, or frame-works that support a particular subset of the problem domain.

For example, additional features could be added to the graphing framework to make it more flexible. In the cur-rent implementation, the label drawer draws only numeric labels, but many graphs use text labels. A better implementation might provide a generic label drawer, with specific derived classes for drawing text labels and numeric labels. It could also be useful to add drawers for other types of graphs, such as line graphs and pie charts. Or perhaps a new framework to support 3-D graphs would be useful.

Also, look for additional ways to make your clients' tasks easier. In some cases, it makes sense to provide special tools with your frameworks. Code generators, CASE tools, and GUI builders can all make programming with frameworks easier, just as they do for traditional software development.

Don't let your frameworks get too big--keep looking for ways to break them down into small, focused frame-works. If they're designed to interoperate, small frameworks are more flexible and can be reused more often.

For example, if you needed support for creating distributed workflow applications, you might initially design a workflow framework with these features:

- Document management
- Rule-based agents
- Store and forward servers
- Networking

A better solution would be to develop separate frameworks for each major function--for example, a document management framework, an agent framework, a client/server framework, and a networking framework. By break-ing down the original workflow framework into a set of small frameworks, the resulting frameworks can be used in other contexts.

It can be difficult to determine when a framework is finished--that's the nature of an iterative process. A simple framework requires fewer iterations than more complex frameworks--another advantage of deve-loping smaller, focused frameworks. Until you actually release your framework, you won't gain any of the benefits.

Generalizing the Design

The concept of prototyping is not unique to framework development, but it is very useful. A common approach is to implement a framework that applies to a specific subset of a larger problem domain, and then rework it to support more general cases. The ET++ SwapsManager project is one example of this approach.

Deriving Frameworks

Once you've developed a general framework that provides a strong architectural base, you can derive additional frameworks that apply to particular problem sets. The overall framework provides generalized components and con-straints to which the derived frameworks conform. Derived frameworks introduce additional components and constraints that support more specific solutions.

Derived frameworks are another method of providing default behavior for your clients. If your framework is made up of a number of abstract classes, you might want to create one or more derived frameworks that provide concrete implementations and additional built-in functionality.

For example, in the Taligent system, the document framework architecture supports the creation of almost all components and documents that will be seen by end users.

This set of frameworks was designed to be very flexible and allow the construction of a wide range of programs. Derived frameworks support a narrower set of applications. For example, a graphical editing framework simplifies the creation of programs that manipulate graphical data. Many standard graphics editing functions such as move, scale, rotate, and group are built into this derived framework.

Documenting the Framework

Code comments and documentation are a necessary part of any programming project, but they are especially important when you're developing frameworks. If other developers don't understand your framework, they won't use it.

Learning how to use someone else's framework can be difficult. As a framework developer, you have to provide enough information so that your clients understand how to use the framework to produce the solution they want.

Make it clear what classes can be used directly, what classes must be instantiated, and what classes must be overridden. Clients are interested in solving particular problems - the details of the framework implementation are not important.

It is best to provide a variety of documentation:

- Sample programs
- Diagrams of the framework architecture
- Descriptions of the framework
- Descriptions of how to use the framework

At a minimum, provide sample programs. In the process of developing and testing your framework, you have to develop applications that use your framework--often these can provide a foundation set of samples. Try to provide a variety of samples that demonstrate how to use the framework in different contexts--a well-rounded set of sample programs is invaluable to clients learning your framework. Consider designing the sample programs so that they show a progression of the architectural features of the framework.

Managing Change

Frameworks evolve--especially as your understanding of the problem domain and number of clients grow. However, once you release a framework for client use, you need to limit the changes--a constantly changing framework is difficult, if not impossible, to use.

As a general rule, you should:

- Fix bugs immediately
- Add new features occasionally
- Change interfaces as infrequently as possible

During the development of a framework, changes happen much more frequently. Once clients have been using a framework for awhile, you might still need to make changes that impact their work.

When you do update a framework, minimize the impact on your clients. It's better to add new classes instead of changing the existing class hierarchy, or to add new functions instead of changing or removing existing functions.

Give your clients advance notice of framework updates and allow time for them to adapt to the changes. One approach is to add the new and changed classes in an interim release and flag the old ones with obsolete warnings. This way, your clients are warned before their code is broken that they are using classes that are going to change.

SUMMARIZING THE FRAMEWORK DEVELOPMENT PROCESS

Developing frameworks allows you to solve a problem once and then reuse that solution any time you face a similar problem. After you've captured domain expertise in a framework, you and others can reuse that framework, saving time and money both in program development and in maintenance. To maximize this benefit, develop a collection of frameworks that you can select from to construct each new program.

You can categorize frameworks by the types of problems the frameworks solve or by the framework structure. The frameworks that you build will likely be domain frameworks, or possibly application frameworks.

Maximizing Framework Benefits

For you and your clients to get the maximum benefit from a framework, the framework should:

- Provide as much built-in functionality as possible
- Minimize the potential for client errors
- Follow standard design and coding guidelines
- Be extensible
- Be portable

Your goal as a framework designer is to get other developers to use your frameworks. If they don't understand how your framework works and how to use it, they'll develop their own solution. You must strike a balance between the simplicity of the client interface and the power of the framework. When in doubt, err on the side of making it simpler for your clients to use the framework, even if it makes implementing the framework more difficult.

Developing Frameworks

When you develop frameworks:

- Derive frameworks from existing problems and solutions
- Develop small focused frameworks
- Build frameworks using an iterative process driven by client participation and prototyping
- Treat frameworks as products--provide documentation and support and plan for distribution and maintenance

Following these guidelines can help you build frameworks that your clients need, that they can use, and that you can support and maintain.

When you begin to develop a framework, you make many small iterations on the design and initial implementation. As the framework matures and your clients begin to rely on it, the changes become less frequent and you move into a much larger maintenance cycle. You can divide this process into four general tasks:

Identify and characterize the problem domain

- Outline the process
- Examine existing solutions
- Identify key abstractions
- Identify what parts of the process the framework will perform
- Solicit input from clients and refine your approach

Define the architecture and design

- Focus on how clients interact with the framework:
  What classes does the client instantiate?
  What classes does the client derive?
  What functions does the client call?
- Apply recurring design patterns
- Solicit input from clients and refine your approach

Implement the framework

- Implement the core classes
- Test the framework
- Ask your clients to test the framework
- Iterate to refine the design and add features

Deploy the framework

- Provide documentation in the form of diagrams, recipes, and especially sample programs
- Establish a process for distribution
- Provide technical support for clients
- Maintain and update the framework

RECOMMENDED READING

The experiences of others are a great source of information and inspiration--you are encouraged to read about what other groups are doing, as well as publish articles about your own endeavors.

The following references include standard object-oriented design references, new publications, and articles about frameworks from a variety of periodicals. Many of the examples included in this paper are based on information in the articles listed here.

PUBLICATIONS

Beck, Kent. "Patterns and Software Development." *Dr. Dobb's Journal* 19, no.2 (February 1994): 18.

Beck, Kent and Johnson, Ralph. "Patterns Generate Architectures." *European Conference on Object-Oriented Programming* (1994).

Birrer, Andreas and Eggenschwiler, Thomas. "Frameworks in the Financial Engineering Domain: An Experience Report." European Conference on Object-Oriented Programming (1993): 21-35.

Booch, Grady. "Designing an Application Framework." *Dr. Dobb's Journal* 19, no.2 (February 1994): 24.

Booch, Grady. *Object-Oriented Analysis and Design With Applications*. Redwood City, CA: Benjamin/Cummings,

1994.

Campbell, Roy; Islam, Nayeem; Raila, David; and Madany, Peter. "Designing and Implementing 'CHOICES': an Object-Oriented System in C++." *Communications of the ACM* 36, no.9 (September 1993)

Coad, Peter. "Object-Oriented Patterns." Communications of the ACM 35, no.9 (1992): 152.

Eggenschwiler, Thomas and Gamma, Erich. "ET++ SwapsManager: Using Object Technology in the Financial Engineering Domain." OOPSLA `92 *Conference Proceedings*, ACM SIG Notices 27, no.10 (1992): 166.

Frameworks: The Journal of Software Development Using Object Technology. Software Frameworks Association.

Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissades, John. "Design Patterns: Abstraction and Reuse of Object-Oriented Design." *European Conference on Object-Oriented Programming* (1993): 406-431.

Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissades, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, to be published October 1994.

Goldstein, Neal, and Jeff Alger. *Developing Object-Oriented Software for the Macintosh*. Reading, MA: Addison-Wesley, 1992.

Mallory, Jim. "TI Software Speeds Semiconductor Production." *Newsbytes* NEW07200011 (July 1993).

Nelson, Carl. "A Forum for Fitting the Task." *IEEE Computer* 27, no.3 (March 1994): 104.

"Semiconductor Industry: New Advanced CIM software from Texas Instruments Expected to Revolutionize Semiconductor Manufacturing." *EDGE: Work-Group Computing Report* 4, no.166 (July 1993): 22.

Shelton, Robert. "The Distributed Enterprise." *The Distributed Computing Monitor* 8, no.10 (October 1993): 3.

Stroustrup, Bjarne. *The C++ Programming Language*. 2d ed. Reading, MA: Addison-Wesley, 1991.

Taligent`s Guide to Designing Programs: Well-Mannered Object-Oriented Design in C++. Addison-Wesley, 1994.

Wong, William. *Plug & Play Programming, An Object-Oriented Construction Kit*. M&T Books, 1993.

SEMINARS

Wilson, Dave. "Designing Object-Oriented Frameworks." Personal Concepts, Palo Alto, CA 1994.
Johnson, Ralph. "How to Design Frameworks." OOPSLA '93 *Tutorial Notes*, 1993.

TALIGENT WHITE PAPERS

A Study of America's Top Corporate Innovators, Taligent, Inc., 1992.
Lessons Learned from Early Adopters of Object Technology, Taligent, Inc., 1993.
Driving Innovation with Technology: The Intelligent Use of Objects, Taligent, Inc., 1993.
Leveraging Object-Oriented Framework, Taligent, Inc., 1993.

PRO 0907

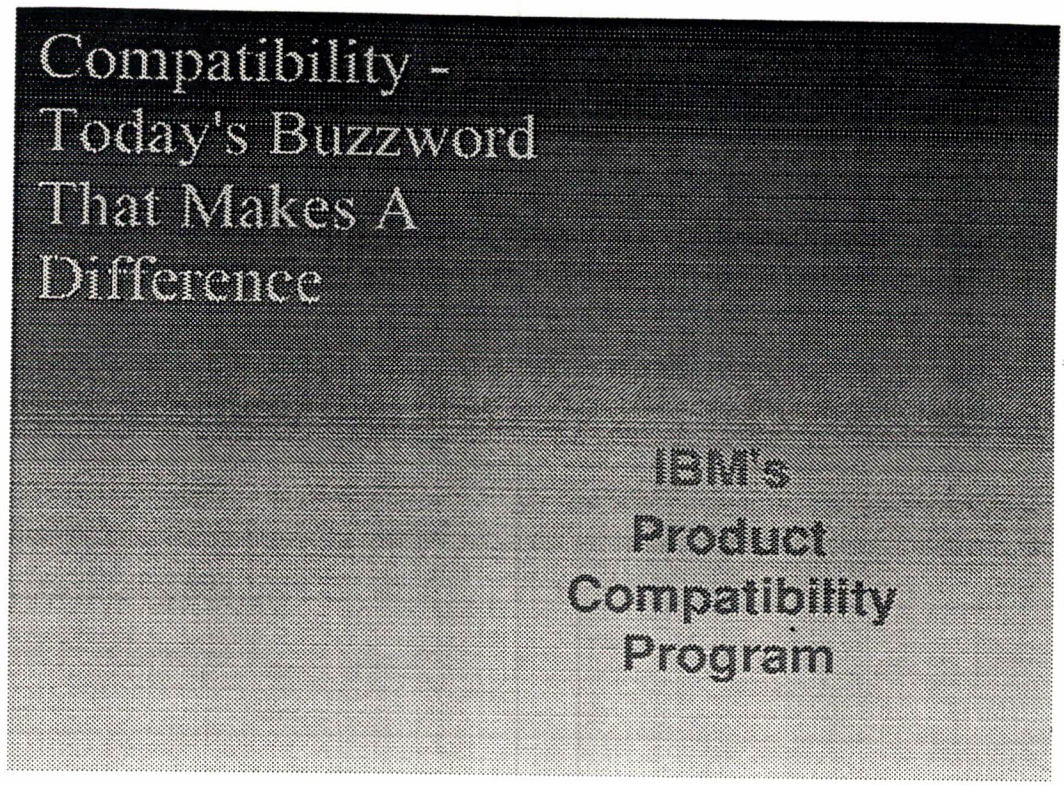
VisualAge — A Powerful New Vision of Programming

No paper available. Please contact speaker

this page intentionally left blank

SYS 0908
Compatability—Today's Buzzword That Makes a Difference

Joyce Batty
IBM



Agenda

Compatibility - Today's Buzzword that Makes a Difference

- Why Compatibility is Important
- IBM's Compatibility Answer
- What's New
- What is IBM's Product Compatibility Program
- How to Increase Marketing Visibility
- How to Find Compatible Products
- A Customized Testing Option

COMPAT.PRS - 1 682F/808/JB

Why Compatibility is Important

- Customers are rightsizing
- New skills required
- Choices of hardware, operating systems, networks, applications are increasing

Why Compatibility is Important

The Problem

- Wasted time and duplicated efforts
- Complicated buying decisions
- Reduced confidence in products
- Slower implementation and usage

LOST BUSINESS

COMPAT.PRS - 2 682F/808/JB

Why Compatibility is Important

Developer Problems

- **Complexity of development**
  - **Select platform operating system, network**
  - **Interoperate with other products, platforms**
  - **Prove the product does what you say**
  - **Find and keep the technical skills**
- **Complexity of marketing**
  - **Establish product awareness & credibility**
  - **Maintain competitive edge**

INCREASED COSTS

Why Compatibility is Important

Customer & Reseller Problems

- **Product choices: Overwhelming**
- **Technical skills must be developed or purchased**
- **Lack of time and resources: Frustrating**

COMPAT.PRS - 3 682F/808/JB

IBM's Compatibility Answer

- Enhance consumer confidence
- Improve buying decisions
- Supplement technical skill and resource availability
- Decrease costs
- Increase marketing products' visibility

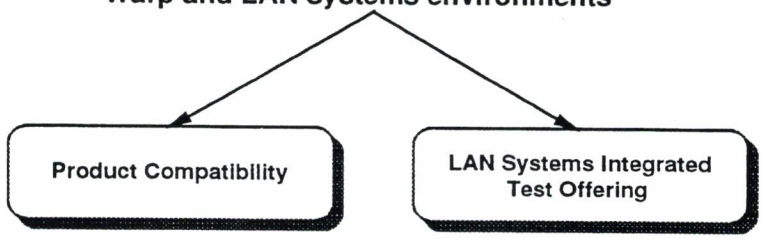
"...customers want assurance that the products they choose will work together. That's why compatibility testing offered by IBM is so important."

Nick Brasier, Product Marketing Manager, Madge Networks

IBM's Compatibility Answer

IBM's Product Compatibility Program

Options designed to increase consumer confidence that products work together in OS/2 ® Warp and LAN systems environments



COMPAT.PRS - 4 682F/808/JB

IBM's Compatibility Answer

Who are These Programs For?

- Consultants
- Developers
- Decision Makers
- Executives
- Integrators
- Project Managers
- Resellers
- Technical Coordinators

People Who Can Thrive on the OS/2 Warp Platform

What's New

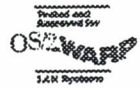
- New Name
- New Compatibility Marks
- New Phone Number (North America)
- New OS/2 Warp Foundation

OS/2® WARP

COMPAT.PRS - 5 682F/808/JB

What's New

= Tested for Compatibility with OS/2 Warp or LAN Server with OS/2 Warp



= Tested for Compatibility/Interoperability with OS/2 Warp in LAN systems environment

= Tested for Compatibility with OS/2 or LAN Server

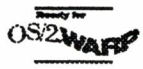
= Tested for Compatibility/Interoperability with LAN systems

IBM's Product Compatibility Program

- Product Compatibility
- LAN Systems Integrated Test Offering

COMPAT.PRS - 6 682F/808/JB

IBM's Product Compatibility Program



- **Ready for OS/2 Warp and Ready for OS/2 Warp LAN Server**
  - Simpler environments
  - Less complicated product focus
  - Compatibility focus
- **Tested and Approved for OS/2 Warp LAN Systems**
  - Complex SYSTEMS environment
  - Industrial strength product focus
  - More than compatibility

IBM's Product Compatibility Program

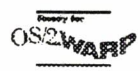
Ready for OS/2 Warp -- Hardware

Compatibility

- Worldwide OS/2/Warp hardware compatibility
- IBM test or self-test
- Included in IBM's OS/2/Warp Hardware Compatibility List

Process

- Information: 1-800-426-4579 (US)
  Int'l.: 1-708-296-6767
- Choose: IBM or self-test
- Sign agreement to use compatibility mark



COMPAT.PRS - 7 682F/808/JB

IBM's Product Compatibility Program

Ready Compatibility - Software

Ready for OS/2 Warp

- Software compatibility: OS/2 Warp
- Self-test
- Stand-alone applications

Ready for OS/2 Warp LAN Server

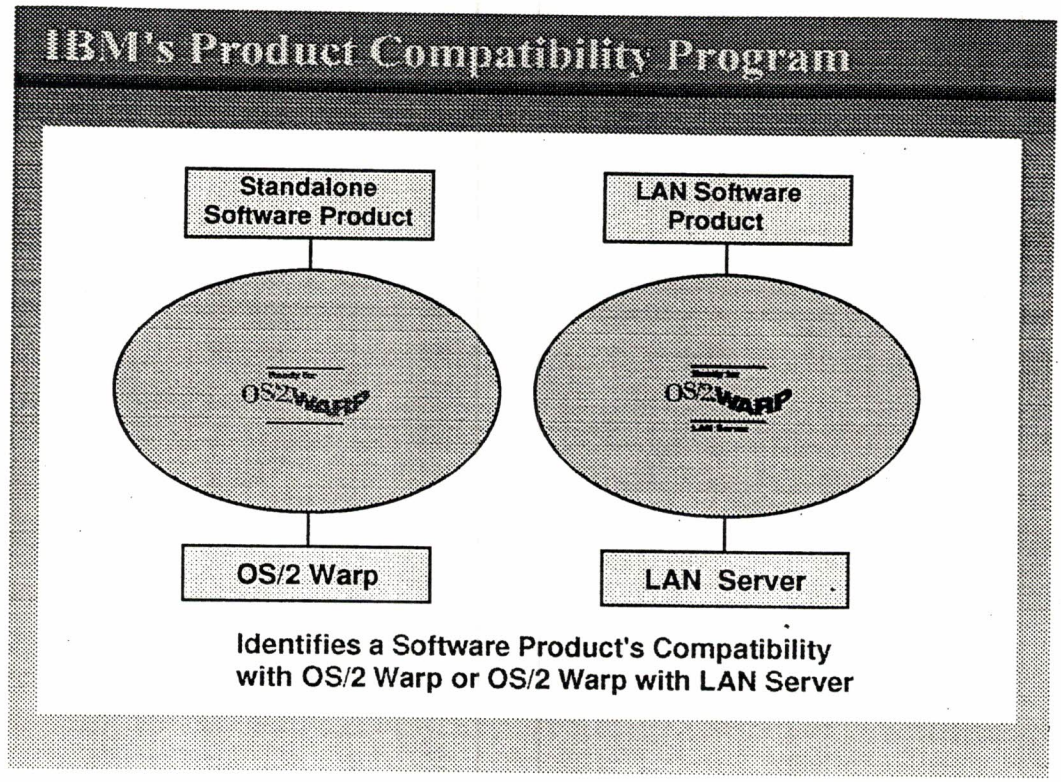
- Software compatibility: OS/2 Warp with LAN server
- Self-test
- LAN applications

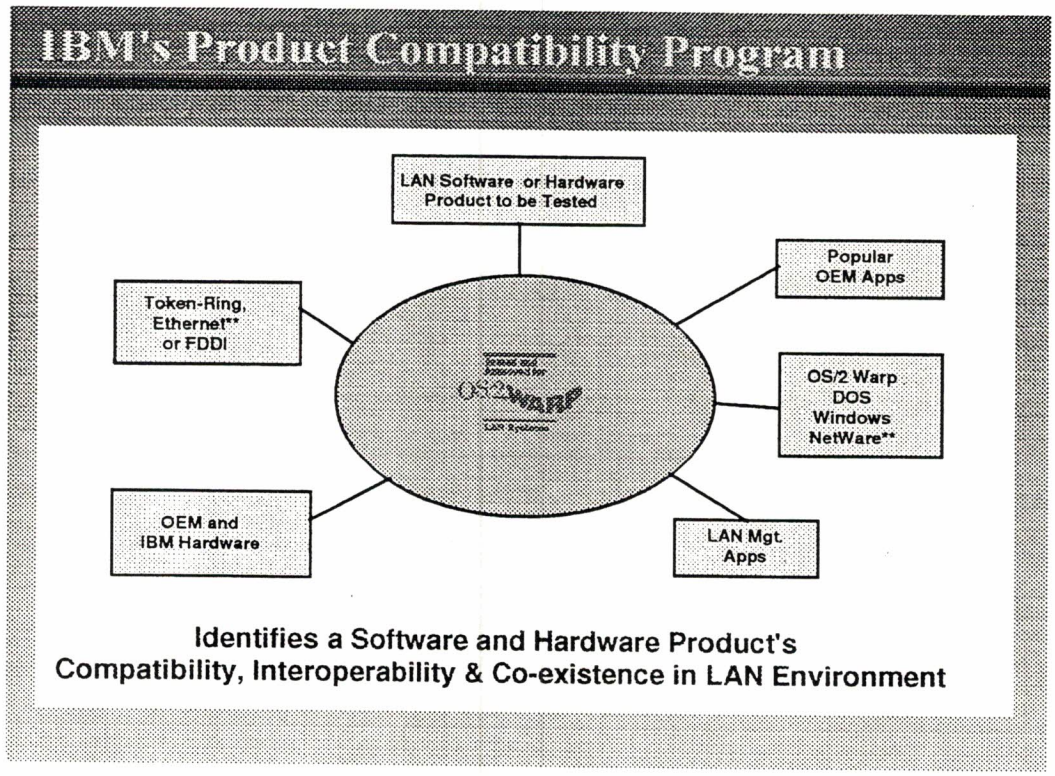
IBM's Product Compatibility Program

Ready Self-Test Process

- Call 1-800-992-4777 to enroll
- Get discounted version of required software (optional)
- Test according to Ready guidelines or have product successfully running in customer environment
- Notify IBM of successful compatibility
- Sign agreement for use of compatibility mark

COMPAT.PRS - 8 682F/808/JB





COMPAT.PRS - 9 682F/808/JB

IBM's Product Compatibility Program

Tested and Approved for OS/2 Warp LAN Systems

- **IBM LAN professionals perform testing**
- **Full testing on LAN platform products in IBM lab**
  - **"Real-World" customer environment**
    - **OS/2 Warp Base**
    - **Token-Ring, Ethernet**, or FDDI**
    - **50% Non-IBM software**
    - **70% Non-IBM hardware**
- **Contractual relationship with developer**
- **Cost: $3,295 to $7,995 per product**

IBM's Product Compatibility Program

Tested and Approved Process

- **Call 1-800-992-4777 for application**
- **Send completed application to IBM**
- **Sign contract for testing**
- **Provide product documentation for IBM testing**
- **Review and approve IBM's test report**
- **Public report published by IBM**
- **Use compatibility mark on deliverables**
- **Enjoy enhanced marketability**

COMPAT.PRS - 10 682F/808/JB

How to Increase Marketing Visibility

- **Establish platform awareness & credibility**
  - **Use and visibility of compatibility mark**
  - **Discounted advertising**
  - **Public distribution of compatible product lists and test reports**

"IBM's LAN Compatibility Program has helped us in a very important area--namely, the bottom line. ...the program allowed us to certify our applications for a popular environment inexpensively and efficiently, ...and has generated hundreds of leads and consequential sales."

-- John Frances, V.P. of Technical Marketing, Halcyon Software, Inc.

How to Find Compatible Products

■ CompuServe** [1]	IBMDESK Forum
■ Computer Reseller News' MAX	1-516-733-6800
■ Fax U.S.: 1-800-IBM-4FAX [+]	DOC: 3241, 3242, 3605
Canada: 1-800-HelpFAX	
■ IBM National Solutions Center	
■ IBM "Sources & Solutions*" Catalog	1-203-452-7704
■ Internet [+]	http://www.austin.ibm.com/pspinfo/compatibility.html
■ OS/2 BBS on Talklink	1-800-547-1283
■ Support on Site** for Networks [+]	1-800-827-7889 1-800-633-8266
■ Developer Connection	

"Some of our customers require compatibility proof to bid. Compatibility testing finds bugs in our products that we miss."

-- Leading Adaptor Company

+ Includes Tested & Approved summary testing reports
[1] Tested and Approved lists and reports only

COMPAT.PRS - 11 682F/808/JB

IBM's Compatibility Test Programs

- Product Compatibility

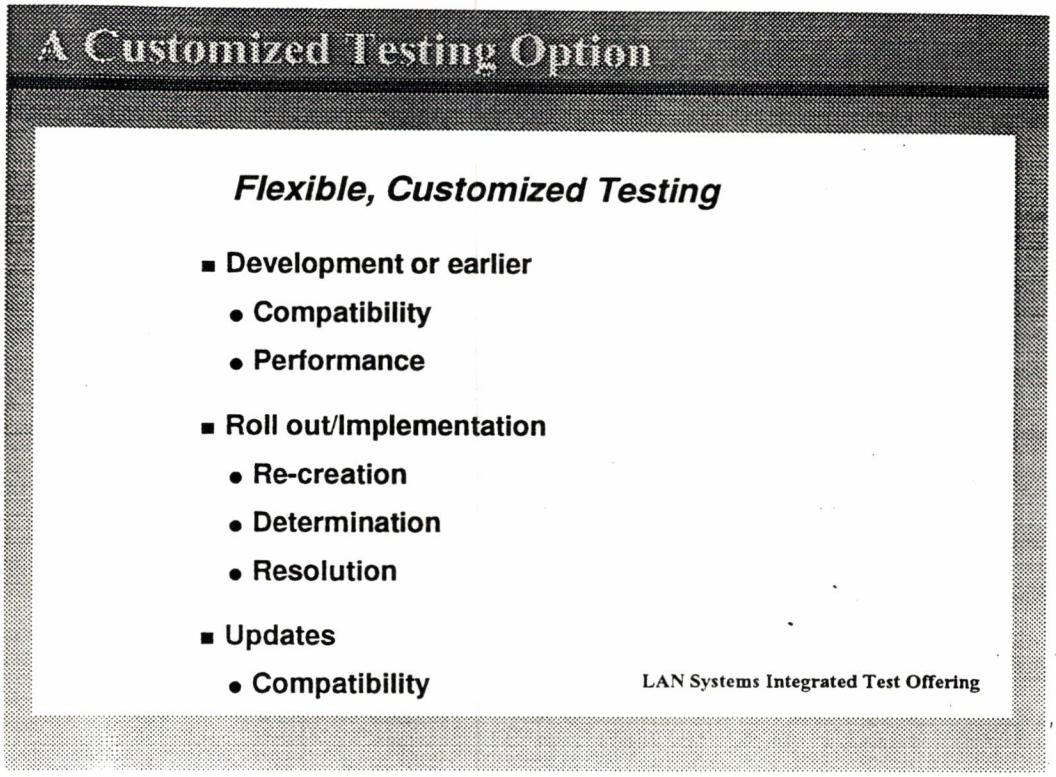
- LAN Systems Integrated Test Offering

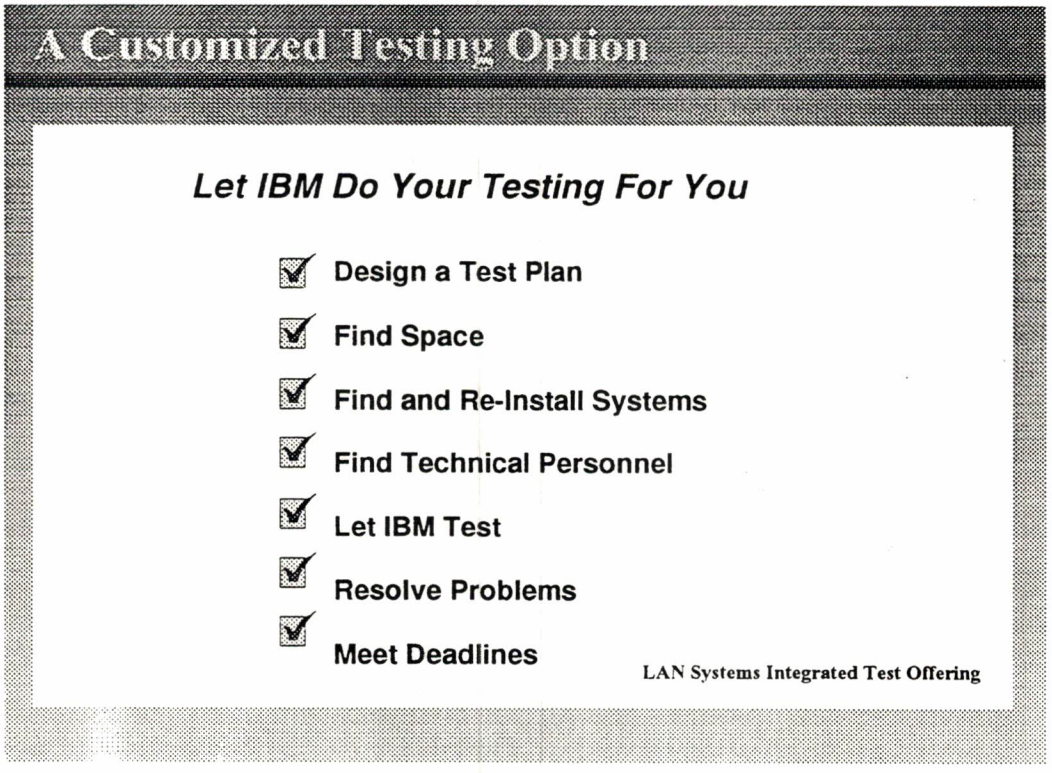
A Customized Testing Option

LAN Systems Integrated Test Offering

A Customized Testing Option designed to test products or upgrades in YOUR LAN systems environment

COMPAT.PRS - 12 682F/808/JB





COMPAT.PRS - 13 682F/808/JB

A Customized Testing Option

Process

1. Call 1-800-992-4777 (U.S. and Canada) or FAX 1-512-838-2473 (International)
2. Receive simple form to describe your needs
3. Send form to IBM
4. Engage with IBM test personnel to evaluate testing requirements
5. Receive test proposal; evaluate; approve contract
6. Observe and assist IBM with test (optional)
7. Review results

LAN Systems Integrated Test Offering

Reactions to Compatibility & Integration Test Programs from IBM

- "We will request that all software products we purchase carry the compatibility mark."
  - *Two Leading Banks*
- "We plan to take our compatibility report on customer sales calls."
  - *Leading Computer Manufacturer*
- "We feel our clients have more confidence in our product because of IBM compatibility."
  - *Philip Lieberman, Owner, Lieberman and Associates, Design and Engineering Group*
- "Glad to see integration and compatibility. How much should I add to my budget to cover this service?
  - *Leading Insurance Company*

COMPAT.PRS - 14 682F/808/JB

Move Ahead Now

Developers

- Enroll in Compatibility Program
  - 1-800-992-4777 North America
    FAX: 1-512-838-2473 (International)
  - Contact local IBM representative in country

Integrators, Resellers, Decision Makers, et.al.

- Look for compatibility marks for purchasing decisions
- Consult bulletin boards and other resources to find lists of compatible products
- Call 1-800-992-4777 for customized integration testing assistance

Trademarks

The following terms, denoted by an asterisk (*) in this presentation, are Trademarks or Registered Trademarks of the IBM Corporation in the United States and/or countries:

IBM CORPORATION	— EXTENDED SERVICES EXTENDED SERVICES FOR OS/2 IBM, NETVIEW, OS/2, PS/2, PS/VALUEPOINT IBM "SOURCES & SOLUTIONS" CATALOG OS/2 WARP

The following terms, denoted by a double asterisk (**) in this presentation, are Trademarks or Registered Trademarks of other companies:

ACER AMERICA CORPORATION	— ACER
ADVANCED MODULAR SOLUTIONS, INC.	— AMS
ASHTON-TATE CORPORATION	— DBASE
AST RESEARCH INC.	— AST
BORLAND INTERNATIONAL, INC.	— PARADOX, QUATTRO PRO
COMPAQ COMPUTER CORPORATION	— MAX
COMPUADD CORPORATION	— COMPAQ
COMPUSERVE INCORPORATED	— COMPUADD
COMPUTER RESELLER NEWS (MAX)*	— COMPUSERVE
DELL COMPUTER CORPORATION	— DELL
DIGITAL EQUIPMENT CORPORATION	— DEC
GATEWAY 2000	— GATEWAY 2000
INTERNATIONAL COMPUTERS, LTD.	— SYSTEMWISE
LOTUS DEVELOPMENT CORPORATION	— LOTUS, cc:MAIL, 1-2-3, FREELANCE NOTES
MICROSOFT CORPORATION	— MICROSOFT, MS-DOS, WINDOWS
NCR CORPORATION	— NCR
NEC CORPORATION	— NEC
NOVELL, INC.	— NETWARE
PACKARD BELL ELECTRONICS, INC.	— PACKARD BELL
SIEMENS NIXDORF AG	— SIEMENS NIXDORF
SOFTWARE PUBLISHING CORPORATION	— HARVARD GRPHICS
WORDPERFECT CORPORATION	— WORDPERFECT
XEROX	— ETHERNET
ZIFF COMMUNICATIONS	— SUPPORT ON SITE, COMPUTER LIBRARY

COMPAT.PRS - 15 682F/808/JB

SYS 0909

Education and the Evolution of OS/2

Feite Kraay

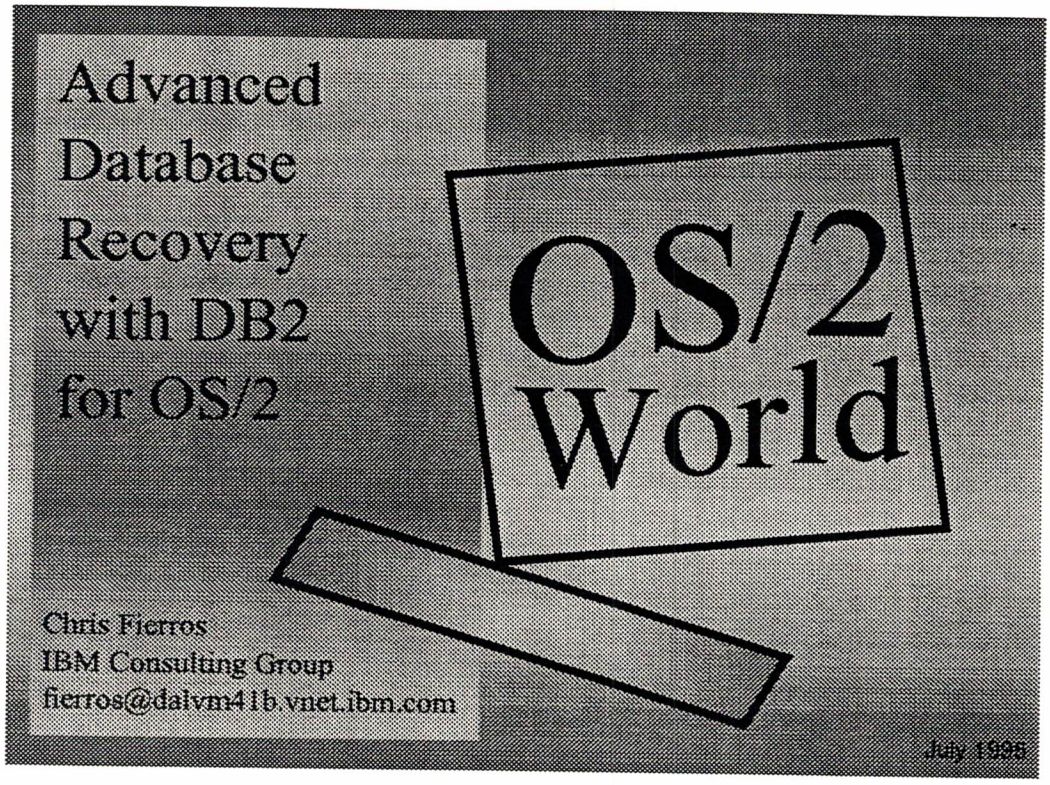
One Up Corporation
Dallas, Texas

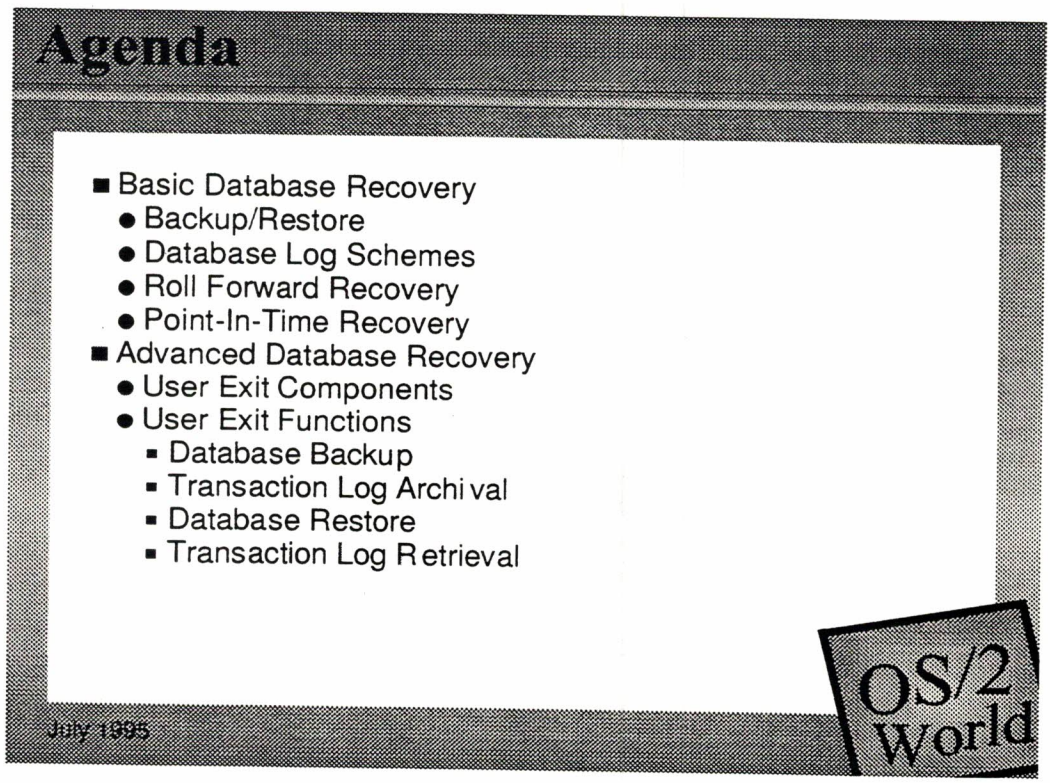
No paper available. Please contact speaker

this page intentionally left blank

CS 1009
Advanced Databases Recovery with DB2/2

Chris Fierros
IBM


Advanced
Database
Recovery
with DB2
for OS/2
OS/2
World
Chris Fierros
IBM Consulting Group
fierros@dalvm41b.vnet.ibm.com
July 1995


Agenda
Basic Database Recovery
Backup/Restore
Database Log Schemes
Roll Forward Recovery
Point-In-Time Recovery
Advanced Database Recovery
User Exit Components
User Exit Functions
Database Backup
Transaction Log Archival
Database Restore
Transaction Log Retrieval
July 1995
OS/2
World

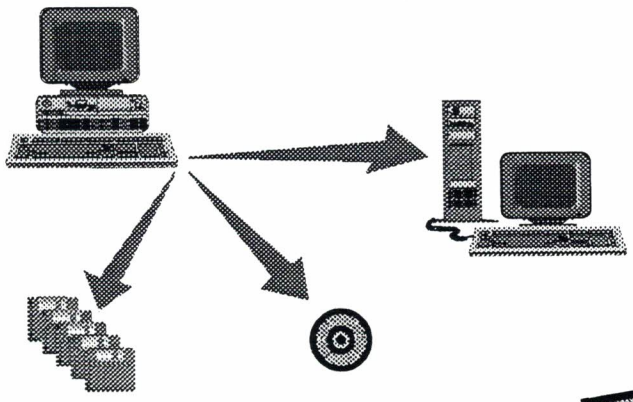
Notes:

- Product Names
  - Database Manager (DBM) EE and ES Versions 1.x
  - DB Manager (DBM) EE and ES Version 1.x
  - Database 2 OS/2 (DB2/2) Versions 1.x
  - DB2 for OS/2 (DB2/2) Versions 2.x
- Database Recovery = Backup/Restore/Rollforward
- Database Logs = Transaction Logs = Log Files

OS/2 World

July 1995

Database Recovery Media



OS/2 World

July 1995

Database Backup Procedure

- Invoked via Database Backup API
  - Recovery Tool
  - Command Line Interface (CLI)
  - C, COBOL, Rexx
- Requires System or DB Adm Level Authority
  - User Logon or Process Logon
- Only Local Databases
- Redirected Target Drive (LAN)
- Supports "changes only" option
- Supports "with quiesce" option

July 1995

OS/2 World

Database Restore Procedure

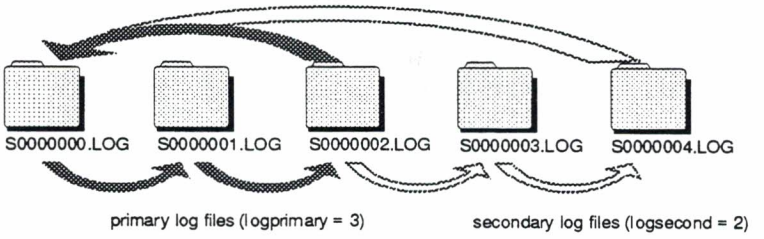
- Invoked via Restore API
  - Recovery Tool
  - Command Line Interface (CLI)
  - C, COBOL, Rexx
- Requires System Level Authority
  - User Logon or Process Logon
- Only Local Databases
- Redirected Source Drive (LAN)
- API must initially be invoked with;
  - callerac = SQLUD_RST_NOROLLFWD
  - callerac = SQLUD_RESTORE ("roll forward pending")
- Returns SQL2505W before database is dropped

July 1995

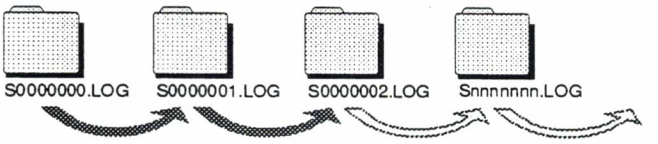
OS/2 World

Database Log Schemes

Circular Log Scheme (log_retain = off)



Linear Log Scheme (log_retain = on)



July 1995

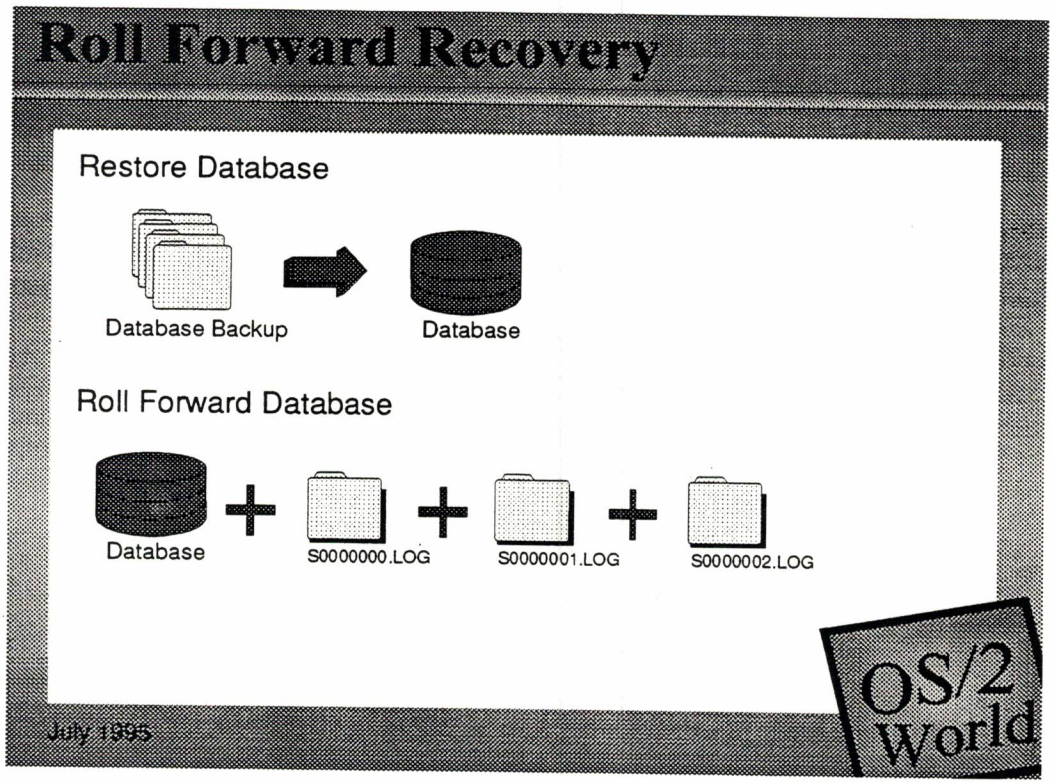
OS/2 World

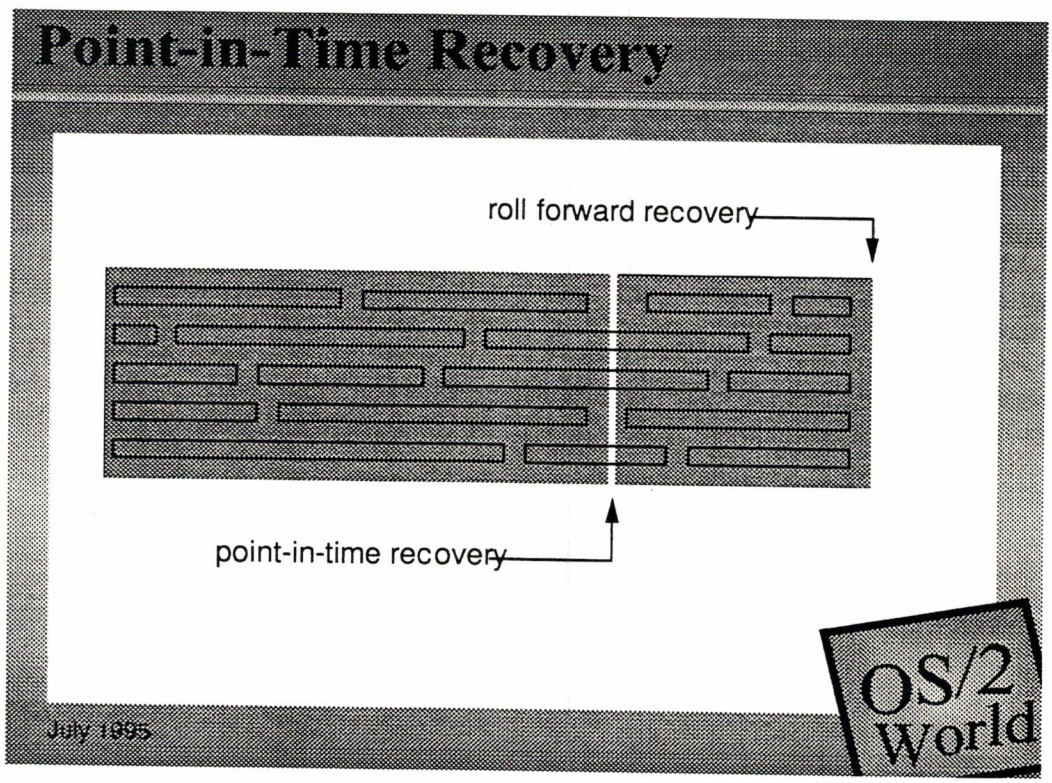
Roll Forward Procedure

- Invoked via Database Roll Forward API
  - Recovery Tool
  - Command Line Interface (CLI)
  - C, COBOL, Rexx
- System Level Authority
  - User Logon or Process Logon
- Roll Forward must be enabled
  - log_retain = on
- Database must be in "roll forward pending" state
  - callerac = SQLUD_RESTORE
  - callerac = SQLUD_CONTINUE
- Supports "point-in-time recovery"

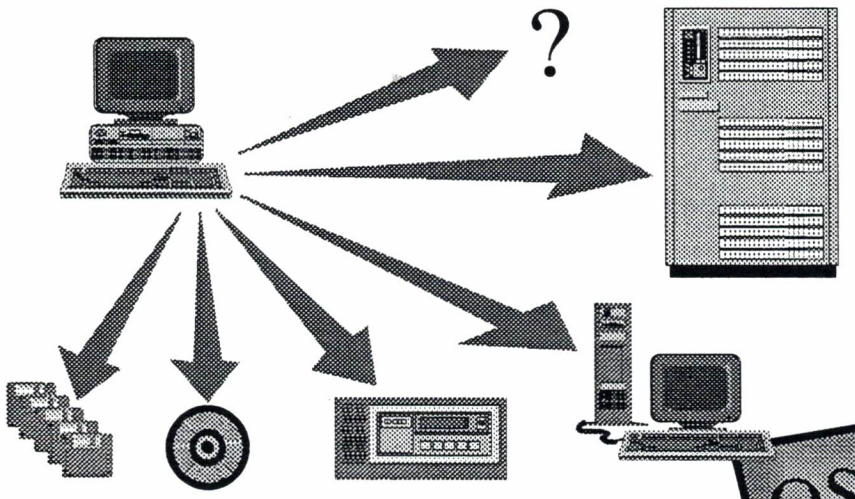
July 1995

OS/2 World


Roll Forward Recovery
Restore Database
Database Backup
Database
Roll Forward Database
Database
S0000000.LOG
S0000001.LOG
S0000002.LOG
July 1995
OS/2 World


Point-in-Time Recovery
roll forward recovery
point-in-time recovery
July 1995
OS/2 World

Advanced Database Recovery Media



July 1995

OS/2 World

Advanced Database Recovery Why?

- Non-Standard Media Device Support
  - tape unit
  - main frame
- Enhanced Functionality
  - backup version control
  - log file management
  - data compression
- Perform Maintenance
  - reorg/runstats
- Additional Fault Tolerance
  - off-line log files

July 1995

OS/2 World

User Exit Program Components

- Requirements
  - Program Name
    - SQLUEXIT.CMD
    - SQLUEXIT.EXE
  - Program Location
    - OS/2 PATH statement
    - C:\SQLUEXIT
  - Receive Parameters
  - Return Codes
- Optional
  - Initialization File
    - SQLUEXIT.INI
  - Log File
    - SQLUEXIT.LOG
  - Audit Log File
    - B0000001.AUD

July 1995

OS/2 World

User Exit Program

- Major Functions
  - Main
  - Backup
    - Backup1
    - Backup2
  - Restore
    - Restore1
    - Restore2
  - Archive
  - Retrieve
- Program Minor Functions
  - Read/Write INI File
    - SQLUEXIT.INI
  - Create/Write Log File
    - SQLUEXIT.LOG
  - Create/Write Audit File
    - B0000001.AUD
  - Read DBM Response Files
    - dbname.BKP
    - dbname.RST
- Invoke OS/2 Commands
  - MD, RD, COPY

July 1995

OS/2 World

User Exit Program

- SQLUEXIT.INI
  - SQLUEXIT_CODE=C:\SQLUEXIT
  - SQLUEXIT_TEMP=D:\TEMP
  - SQLUEXIT_BACK=Z:\BACKUP
  - SQLUEXIT_LOGS=Z:\BACKUP
  - SQLUEXIT_COMP=ALL
  - SQLUEXIT_NEXT=0000001
  - SQLUEXIT_NUMB=0000003

July 1995

OS/2 World

User Exit Program

- SQLUEXIT.LOG

```
=================================================================
SQLUEXIT.EXE Version 1.0          DATE: 03/02/95 TIME: 12:00:00
*****************************************************************
BACKUP     SAMPLE       DATE: 03/02/95  TIME: 12:00:00
BACKUP F: SAMPLE    F:\SQLDBDIR\SQL00001.BKP SAMPLE-755523387 1
CreateTempDir:F:\BACKUP\TEMP RC:00000
...
```

July 1995

OS/2 World

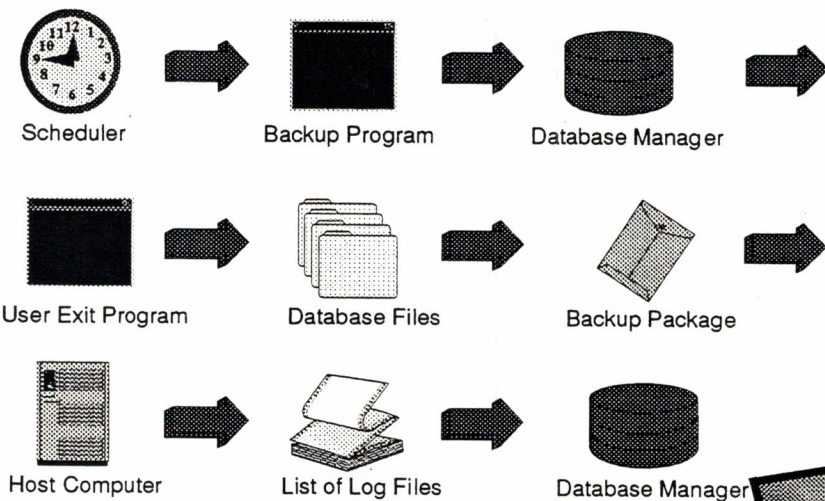
User Exit Program

```
■B000000.AUD
  ●=====================================
  ●SQLUEXIT V1.0     03/02/95      12:00:00
  ●=====================================
  ●
  ●Log File          Log Date      Log Time
  ●============      ========      ========
  ●S0000000.LOG      03/02/95      12:00:00
  ●S0000001.LOG      03/02/95      12:15:00
  ●S0000002.LOG      03/02/95      12:30:00
  ●S0000003.LOG      03/02/95      12:45:00
  ●S0000004.LOG      03/02/95      13:00:00
  ●S0000005.LOG      03/02/95      13:15:00
  ●S0000006.LOG      03/02/95      13:30:00
  ●S0000007.LOG      03/02/95      13:45:00
  ●S0000008.LOG      03/02/95      14:00:00
  ●*************End of Log List**********
```

OS/2 World

July 1995

Database Backup



OS/2 World

July 1995

Database Backup

- Backup Phase 1 - parameters
  - PARMA=BACKUP
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQLDBDIR\SQL00001.BKP
  - PARM4=SAMPLE-731767629
  - PARM5=1

July 1995

Advanced Backup

- Backup Phase 1 - pseudo code
  - get system date time
  - open SQLUEXIT.LOG file
  - write header record to SQLUEXIT.LOG file
  - write user exit parameters to SQLUEXIT.LOG file
  - make temporary directory for database files
  - open dbname.BKP file
  - read dbmname.BKP file
  - copy files listed in dbmname.BKP to temporary directory
  - close SQLUEXIT.LOG file

July 1995

Database Backup

- Phase 2 - parameters
  - PARMA=BACKUP
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQLDBDIR\SQL00001.BKP
  - PARM4=SAMPLE-731767629
  - PARM5=2

July 1995

OS/2 World

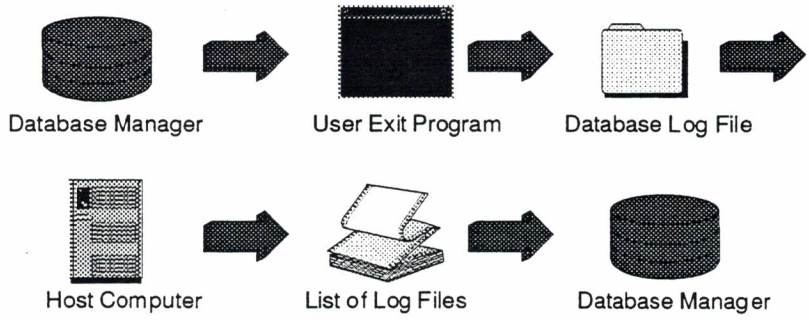
Advanced Backup

- Backup Phase 2 - pseudo code
  - perform Backup Phase 1 tasks
  - zip database files in temporary directory
  - copy B00000000.ZIP file to backup media
  - deletes database files
  - removes temporary directory
  - archives log audit file (B0000000.AUD)
  - creates new log audit file (B0000001.AUD)
  - update SQLUEXIT-NUMB in INI file
  - close SQLUEXIT.LOG file

July 1995

OS/2 World

Transaction Log Archival



July 1995

OS/2 World

Transaction Log Archival

- Archive - parameters
  - PARMA=ARCHIVE
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQL00001\SQLOGDIR\
  - PARM4=S0000000.LOG
  - PARM5=

July 1995

OS/2 World

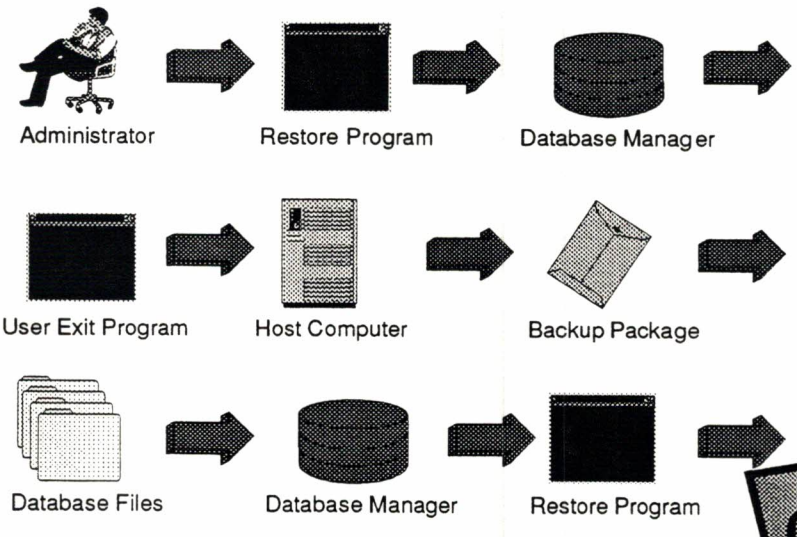
Transaction Log Archival

- Archive - pseudo code
  - get system date and time
  - open SQLUEXIT.LOG file
  - write header record to SQLUEXIT.LOG file
  - write user exit parameters to SQLUEXIT.LOG file
  - copy log file (Snnnnnnn.LOG) to backup media
  - update log audit file (B0000001.AUD)
  - close SQLUEXIT.LOG file

July 1995

OS/2 World

Database Restore



July 1995

OS/2 World

Database Restore

- Restore Phase 1 - parameters
  - PARMA=RESTORE
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQLDBDIR\SAMPLE.RST
  - PARM4=\SQL00001\
  - PARM5=1

July 1995

Database Restore

- Restore Phase 1 - pseudo code
  - get system date and time
  - open SQLUEXIT.LOG file
  - write header record to SQLUEXIT.LOG file
  - write user exit parameters to SQLUEXIT.LOG file
  - make temporary directory for database files
  - get database file(s) from backup media
  - unzip database files in temporary directory
  - copy requested database files to database directory
  - close SQLUEXIT.LOG file

July 1995

Database Restore

- Restore Phase 2 - parameters
  - PARMA=RESTORE
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQLDBDIR\SAMPLE.RST
  - PARM4=\SQL00001\
  - PARM5=2

July 1995

OS/2 World

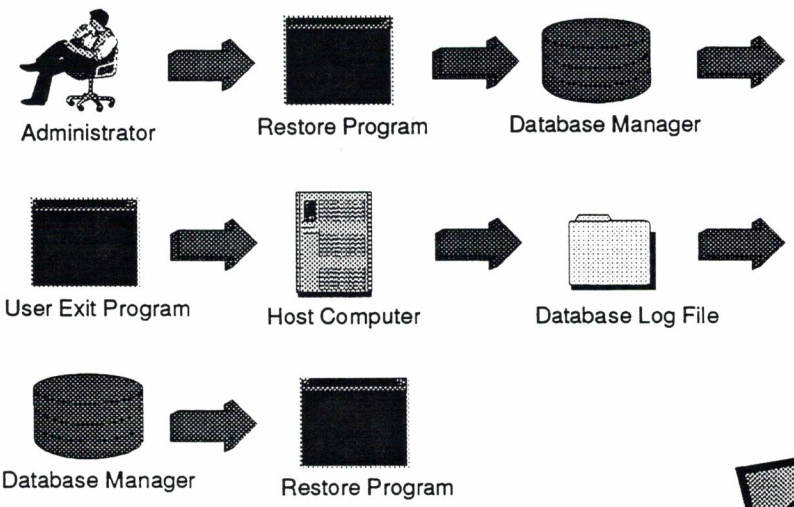
Database Restore

- Restore Phase 2 - pseudo code
  - get system date and time
  - open SQLUEXIT.LOG file
  - write header record to SQLUEXIT.LOG file
  - write user exit parameters to SQLUEXIT.LOG file
  - copy requested database files to database directory
  - delete database files from temporary directory
  - remove temporary directory
  - close SQLUEXIT.LOG file

July 1995

OS/2 World

Transaction Log Retrieval



July 1995

OS/2 World

Transaction Log Retrieval

- Retrieve - parameters
  - PARMA=RETRIEVE
  - PARM1=C:
  - PARM2=SAMPLE
  - PARM3=C:\SQL00001\SQLOGDIR\
  - PARM4=S0000000.LOG
  - PARM5=

July 1995

OS/2 World

Transaction Log Retrieval

- Retrieve - pseudo code
  - get system date and time
  - open SQLUEXIT.LOG file
  - write header record to SQLUEXIT.LOG file
  - write user exit parameters to SQLUEXIT.LOG file
  - get log file from backup media
  - close SQLUEXIT.LOG file

July 1995

How to get Started?

- IBM Database 2 OS/2 Guide, Appendix I
- IBM Database 2 OS/2 "Cookbook" Keeping it in the Family
- User Exit Sample Programs
  - SQLUEXIT.EX1 - SYTRON Corporation's Sytos Plus
  - SQLUEXIT.EX2 - Mountain Corporation's Filesafe
  - SQLUEXIT.EX3 - Maynard Corporation's MaynStream
  - SQLUEXIT.EX4 - OS/2 xcopy

July 1995

LAN 1010
LAN Automated Distribution/2

Bryan Heft
IBM

Agenda

- Introduction
- Overview
- Features & Functions
- Future Enhancements
- Support & Pricing
- Demonstration

LAD/2 Overview

- PM Interface
- Ease of Use - Simplifies Software Distributi
- Saves Time and Resources
- Supports Both CID and Non-CID Products
- Automated Functions

LAD/2 Features & Functions

- LCU Based Only
- Object Based Interface (icons)
- Drag/Drop Capability
- Open Architecture
- Dynamic Client & Product Grouping
- Product Notebooks
- Generic Response File Generator

LAD/2 Features & Functions

- Complete FDISK Partitioning
- Multiple Partition Formatting
- BootManager Front & Back
- Response File Generation
- LCU Command File Generation
  - ► User Ordering Capability

LAD/2 Features & Functions

- Manipulate & Edit System Files
- Manipulate & Edit .INI Files
- Client Boot Diskette Creation
  - ► 1.44Mb & 2.88Mb Support
- Client Data
  - ► Individual Configuration Data
  - ► Installed Products History

Future Enhancements

- NETBIOS Under TCP/IP
- Multiple Server Platforms
- DOS Workstation Support
- DOS Settings Capability
- Remote Administration

Support & Pricing

- Pricing
  - Marketed through ASPO Product Channe
  - 1-800-IBM-3333 ext. IC1000
- Support
  - LAD/2 CFORUM (TALKLINK)
  - Internet

MNC 1004
Communications Manager/2: Advanced Configurations

George Sweeney
IBM

Communications Manager/2 Advanced Configuration

Research Triangle Park, NC

Agenda

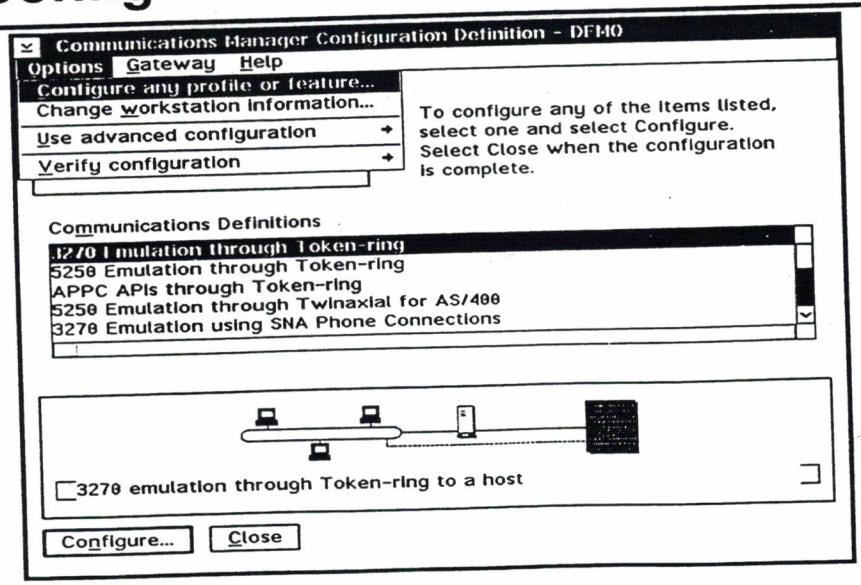
- ► Advanced Configuration Overview
- • 3270 Configuration to multiple hosts
- • Gateway Configuration
- • 5250 Configuration *including async dialup*
- • APPC/APPN Configuration
- • CID Support
- • Dependent LU Requester

IBM Operating System/2™

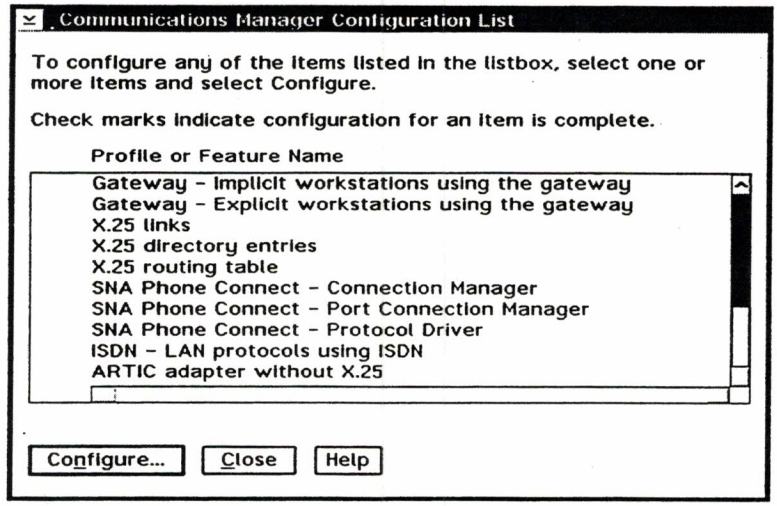
Advanced Configuration

- Required for some configurations
- Select 'Advanced...' pushbutton or 'Options' on menu bar
- Guided profile lists
- Checkmarks

Configuration Definition 'Options'



Configure any profile or feature



Selected from 'Options' on Config Definition

Agenda

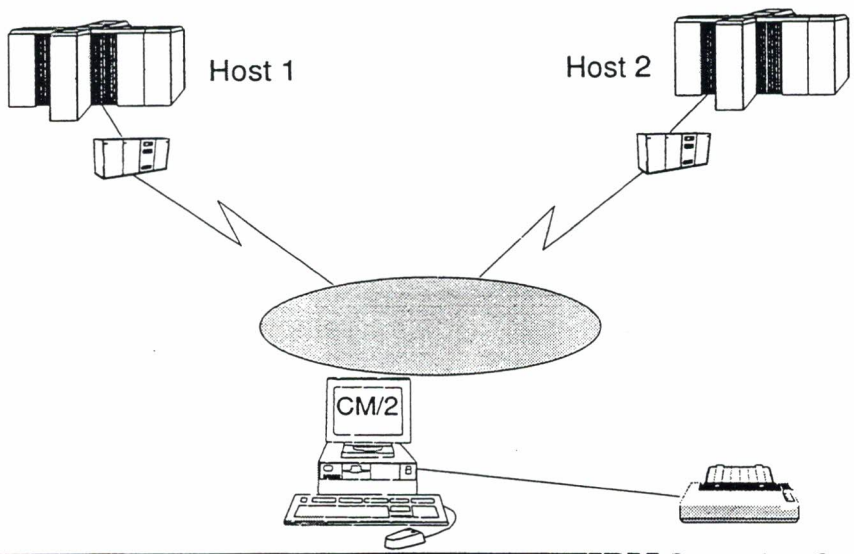
- **Advanced Configuration Overview**
- ► 3270 Configuration to multiple hosts
- **Gateway Configuration**
- **5250 Configuration *including async dialup***
- **APPC/APPN Configuration**
- **CID Support**
- **Dependent LU Requester**

3270 Configuration

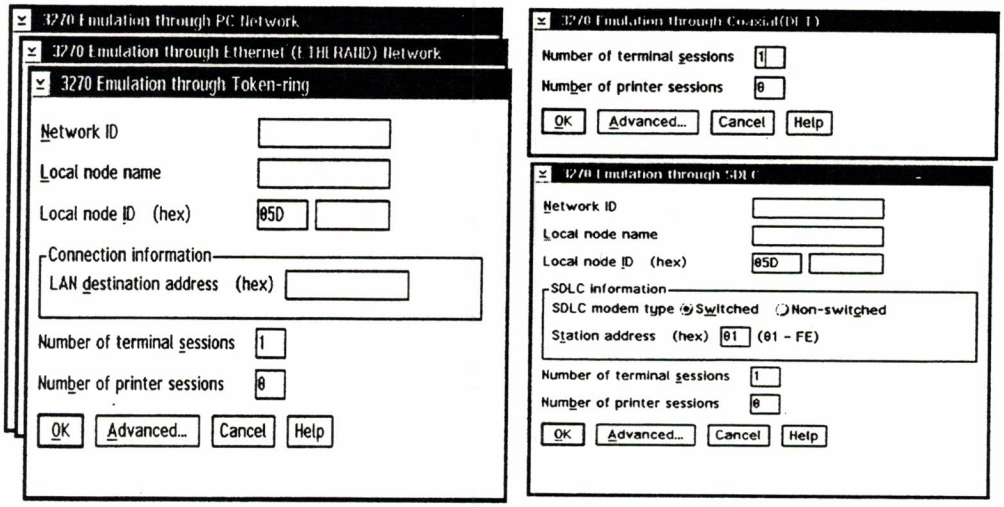
- OBJECTIVE
  - Review quick configuration windows
  - Configure 2 token-ring sessions, each to a different host
  - Configure 1 printer session

- PROCEDURE
  - Type CMSETUP or Select CM Setup ICON
  - Select 'Setup' pushbutton from CMSETUP
  - Choose configuration name
  - Select 3270 emulation through Token-Ring

3270 Network View



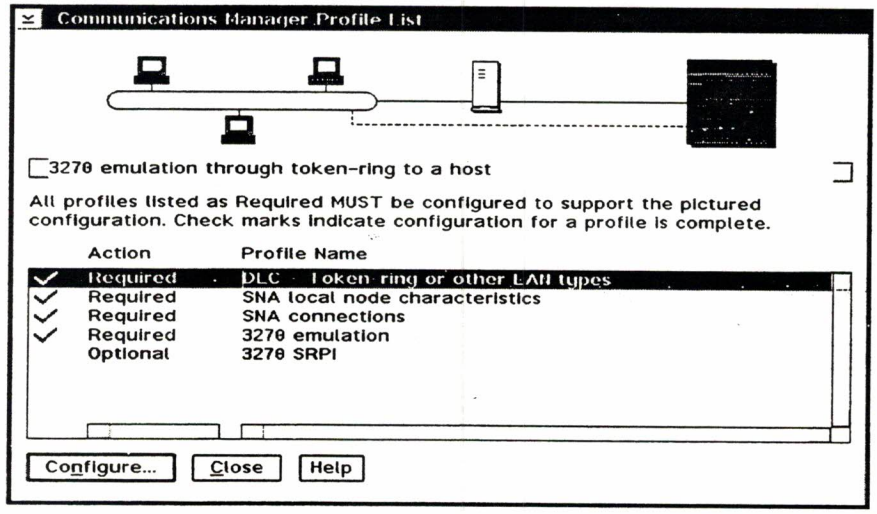
3270 Quick Configuration



Fill in values, select 'Advanced...'

IBM Operating System/2 ™

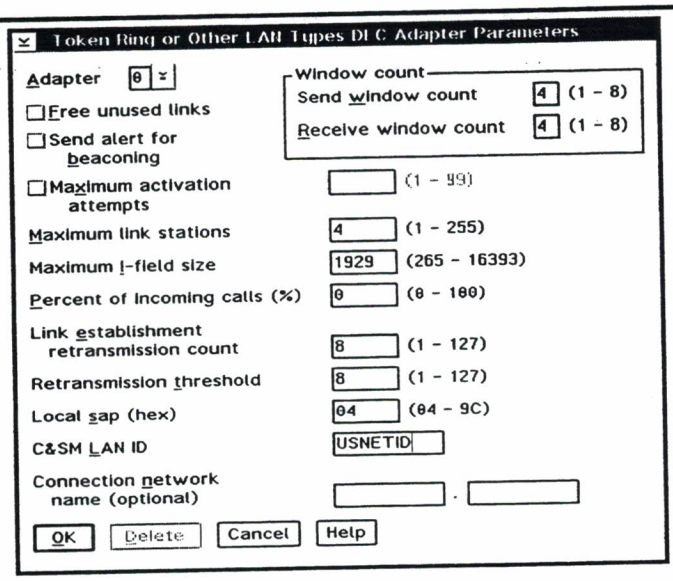
3270 Profile List



Select profile, then 'Configure...'

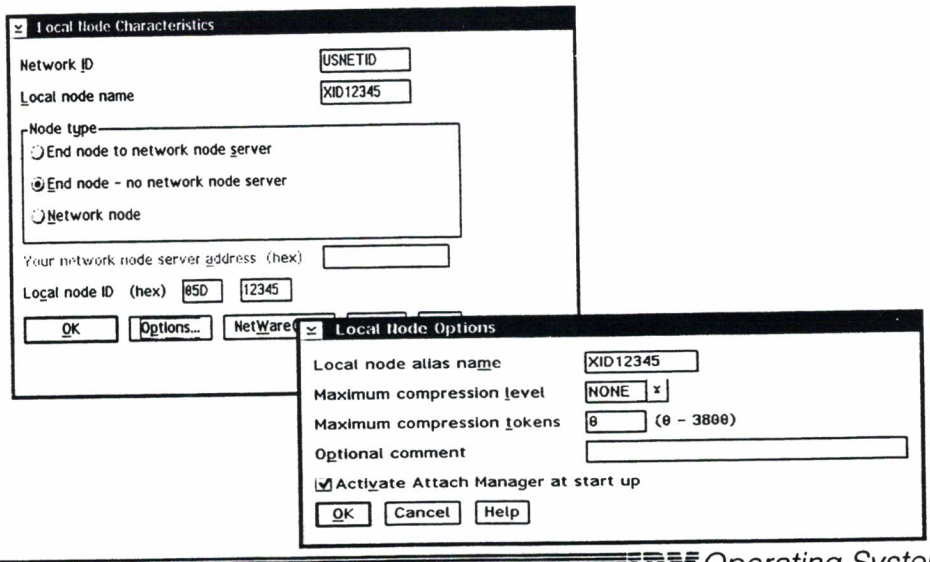
IBM Operating System/2 ™

LAN DLC



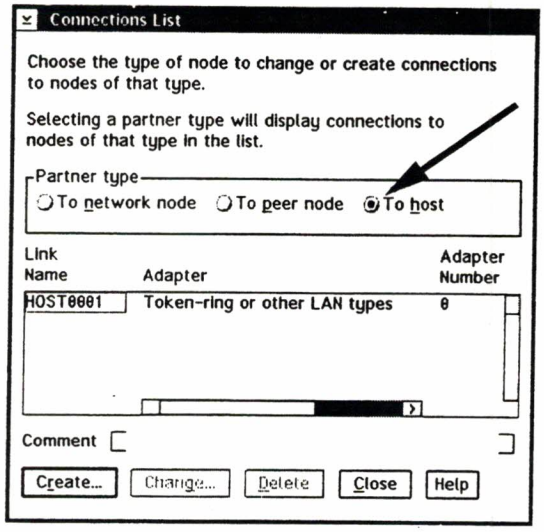
IBM Operating System/2

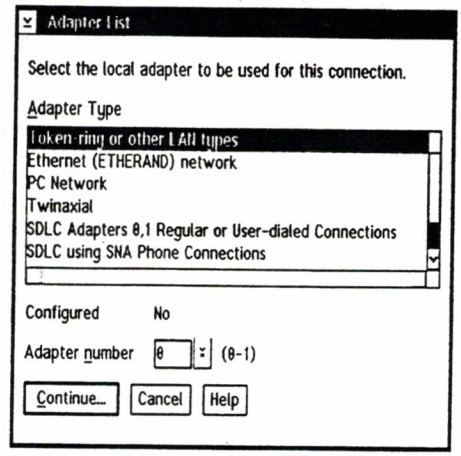
SNA Local Node Characteristics



IBM Operating System/2

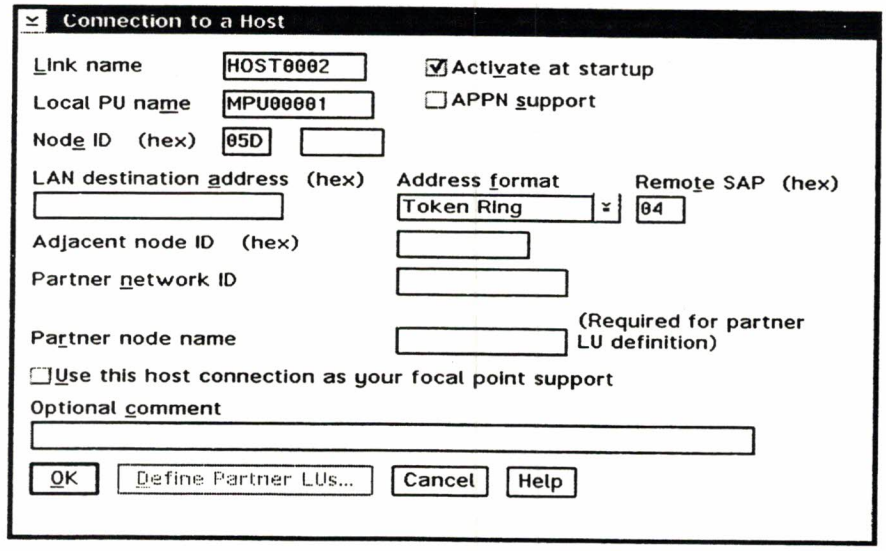
Create a Connection





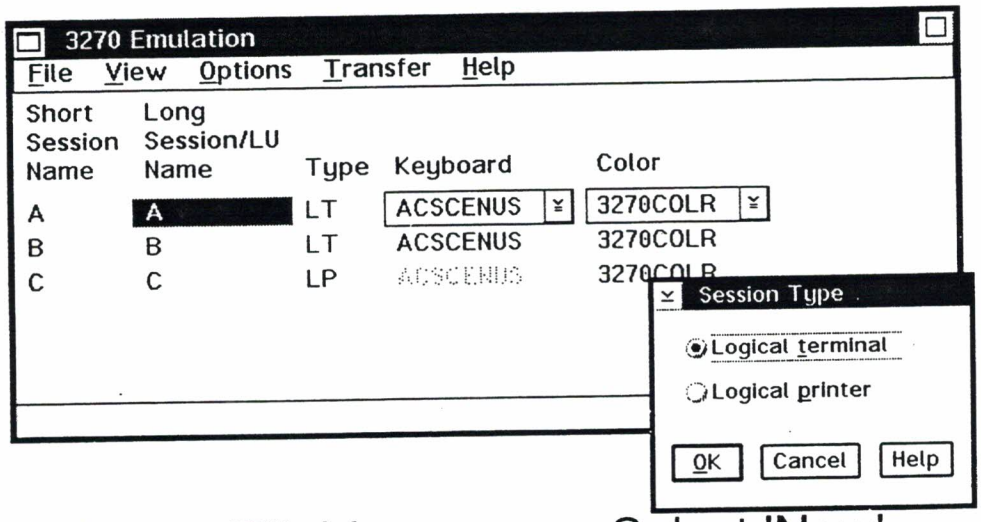
IBM Operating System/2™

Create a Connection...



IBM Operating System/2™

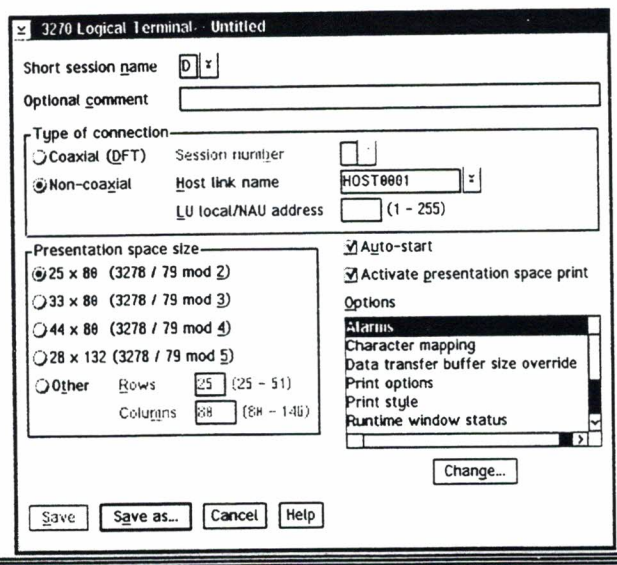
3270 Emulation



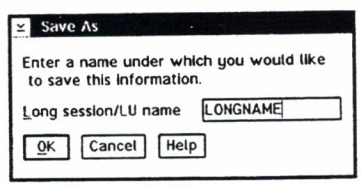
Select 'File' from menu, Select 'New'.

IBM Operating System/2 ™

3270 Emulation...

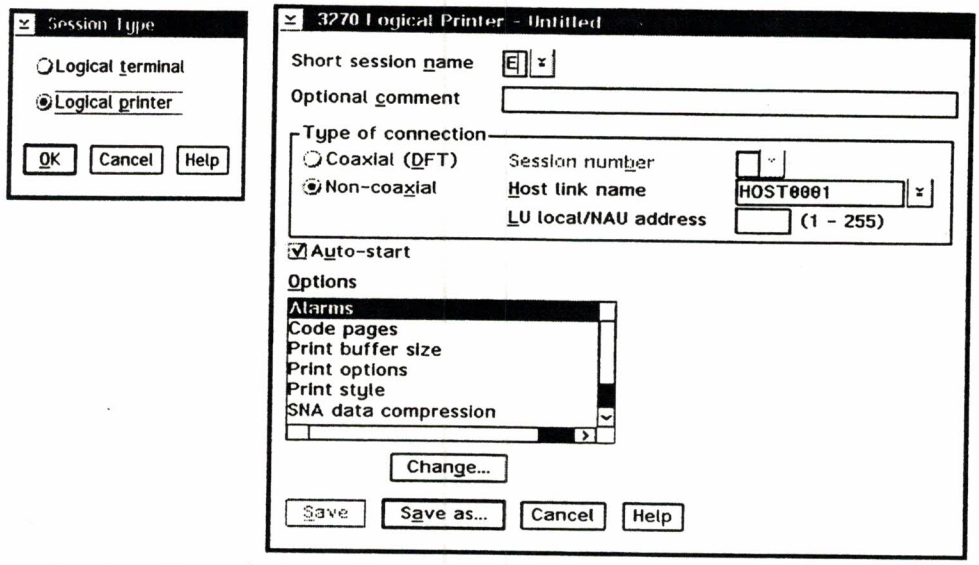


Fill in values.
Select 'Save as...'



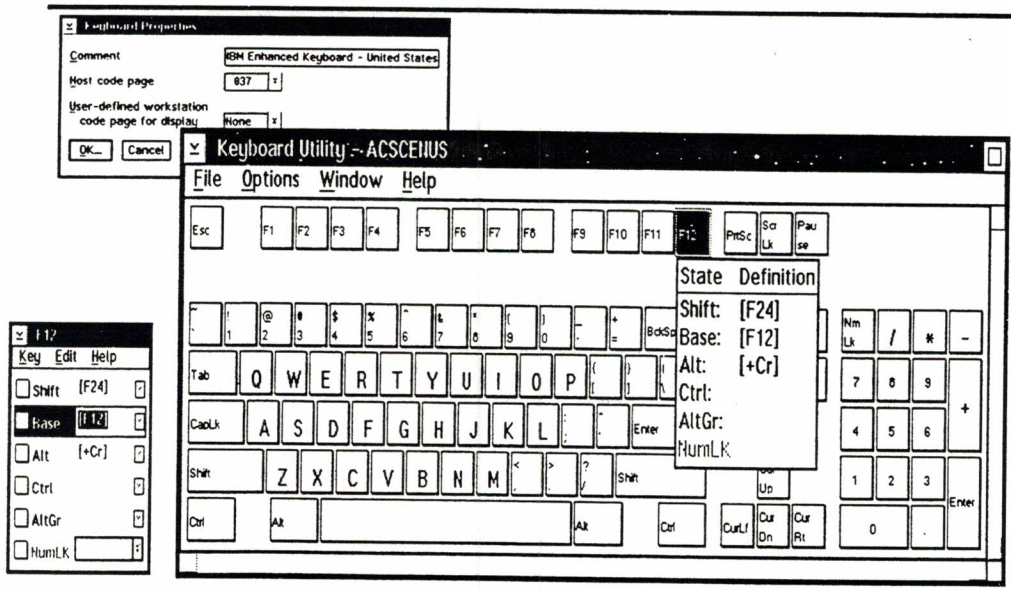
IBM Operating System/2 ™

3270 Logical Printer



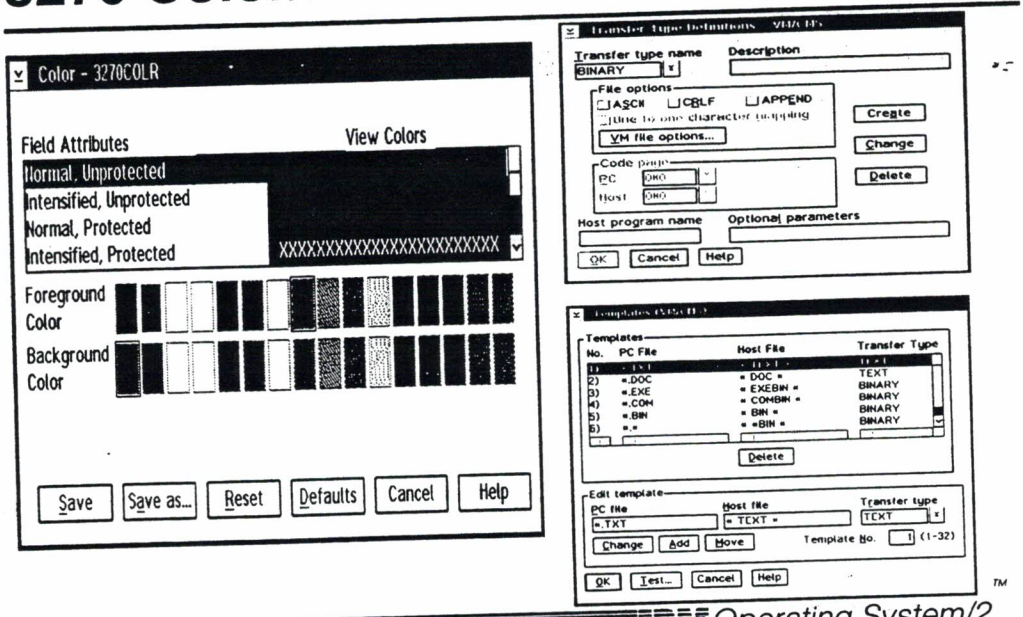
IBM Operating System/2

3270 Keyboard



IBM Operating System/2

3270 Color/File Transfer



IBM Operating System/2

Agenda

- Advanced Configuration Overview
- 3270 Configuration to multiple hosts
- ► Gateway Configuration
- 5250 Configuration *including async dialup*
- APPC/APPN Configuration
- CID Support
- Dependent LU Requester

IBM Operating System/2

Gateway Overview

- Access host resources for downstream workstations
  - up to 5 different hosts (PU's), 254 sessions per PU
- Concentrate host definitions at the gateway
- Workstations access host via different media
- Gateway can also be network node server

- Downstream workstations supported:
  - Communications Manager/2 1.0, 1.1
  - Extended Services
  - Personal Communications/3270 2.0, 3.0, 3.1
  - 3270 Emulation Program Version 3
  - 3270 Workstation Program Version 1.1
  - NS/DOS Version 1.0
  - Other vendor products

IBM Operating System/2™

Gateway Terminology

- Dedicated LU
  - permanent reserved LU from downstream workstation to host LU
- Pooled LU
  - group of LU's from the host
  - downstream workstation use next available LU from the pool

- Explicit Workstation
  - define every downstream workstation at the gateway
- Implicit workstation
  - gateway definition dynamically created

- Dependent LU's supported
  - 3270 emulator
  - LUA applications
  - LU 6.2 dependent LU's

IBM Operating System/2™

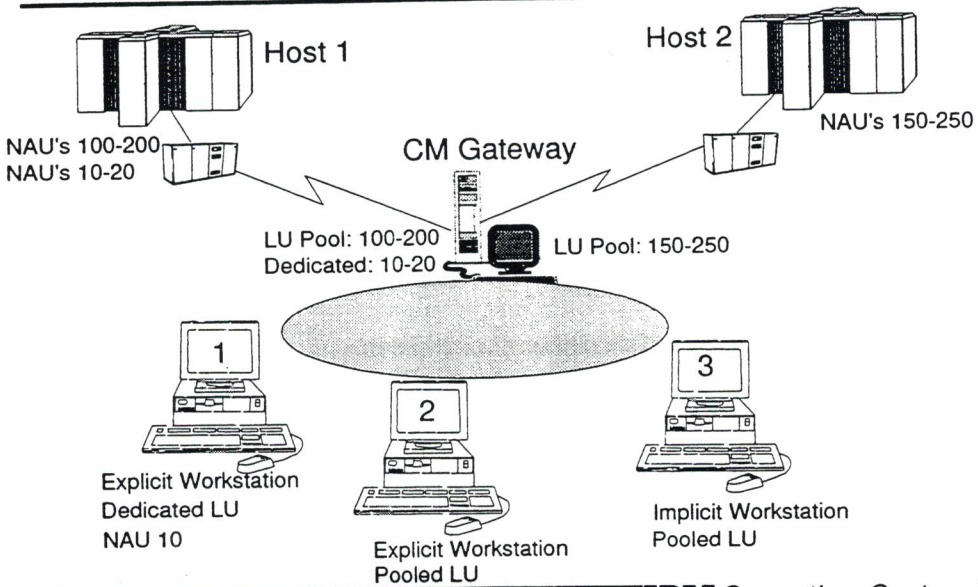
Gateway Configuration

- OBJECTIVE
  - Configure Gateway Host Link
  - Configure Gateway Workstation Link

- PROCEDURE
  - Type CMSETUP or select CM Setup ICON
  - Select 'Setup' pushbutton from CMSETUP
  - Choose configuration name
  - Select 'Gateway' from Menu bar
  - Select 'Configure'

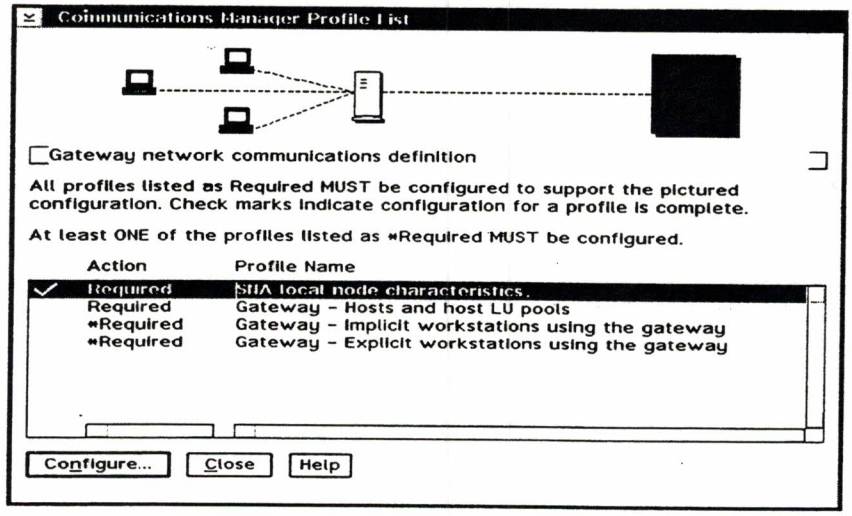
IBM Operating System/2™

Gateway Network View



IBM Operating System/2™

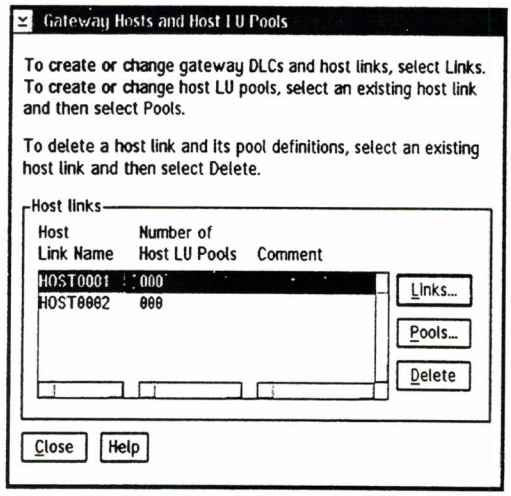
Gateway Profile List

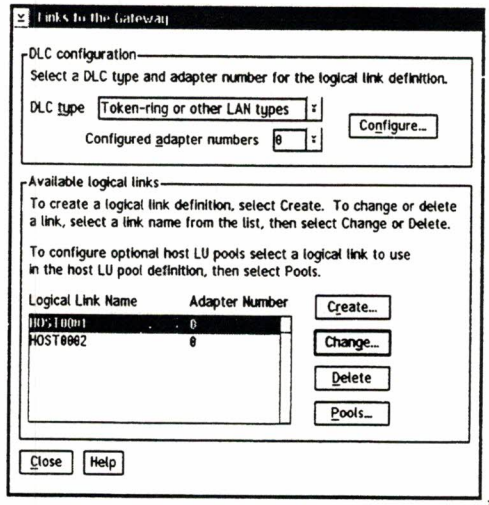


Select profile, then 'Configure...'

IBM Operating System/2

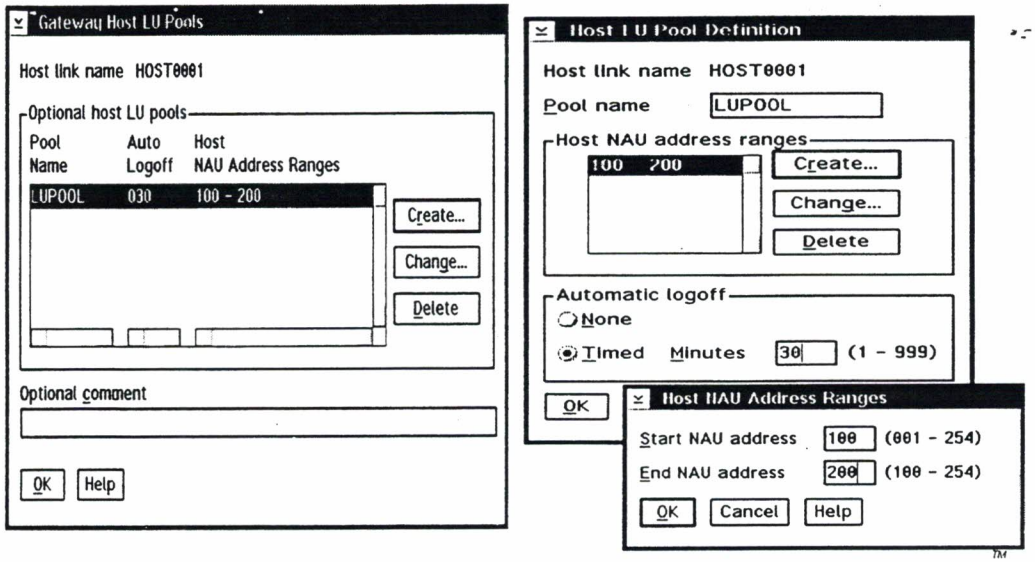
Gateway Hosts and Host LU Pools





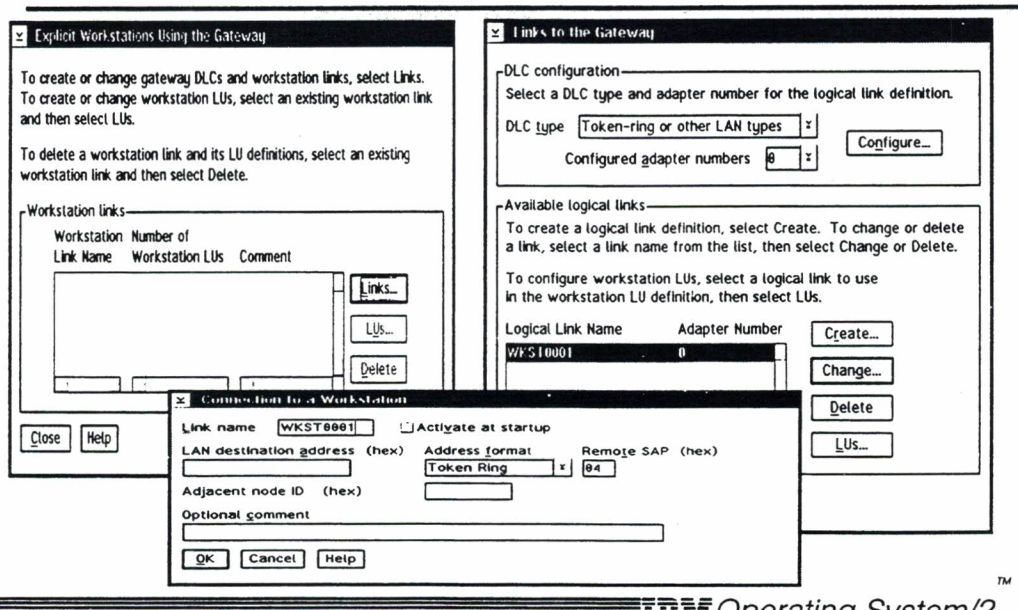
IBM Operating System/2

Gateway Host LU Pool Definitions



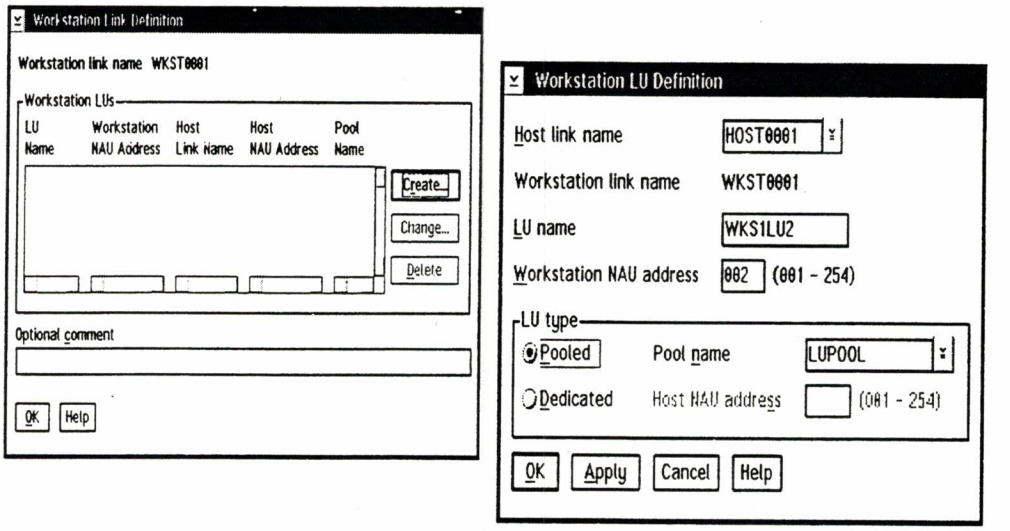
IBM Operating System/2

Explicit Gateway Workstations



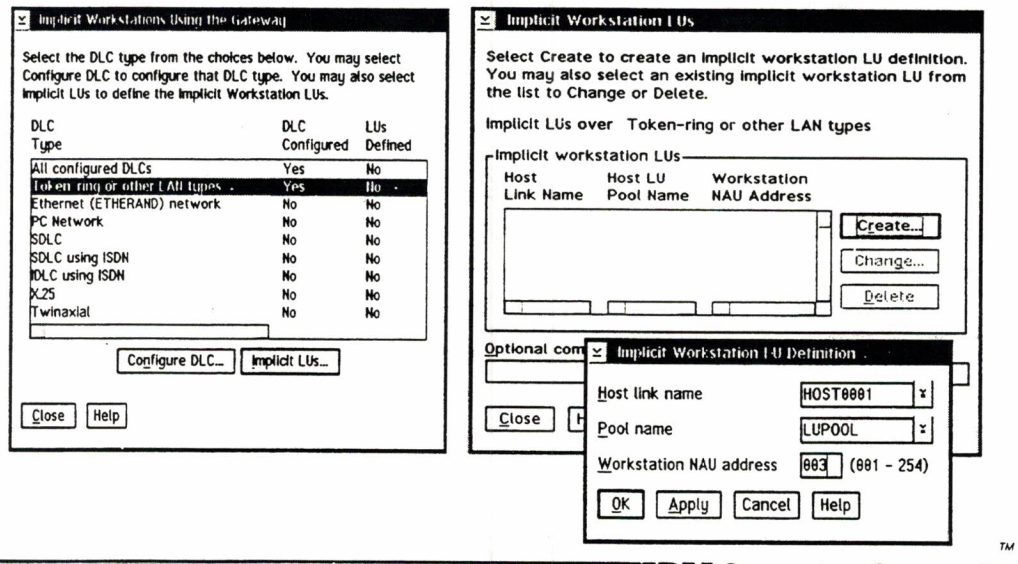
IBM Operating System/2

Explicit Gateway Workstations...



IBM Operating System/2

Implicit Workstations Using the Gateway



IBM Operating System/2

Agenda

- Advanced Configuration Overview
- 3270 Configuration to multiple hosts
- Gateway Configuration
- 5250 Configuration *including async dialup*
- APPC/APPN Configuration
- CID Support
- Dependent LU Requester

IBM Operating System/2™

SNA Phone Connection Overview

- ***Addresses mobile computing environment:***
  - ***use SNA applications from home, traveling, or wherever there is a phone line***
- ***Connectivity supported:***
  - ***SNA applications over asynchronous lines (Switched/Non-switched)***
  - ***SNA applications over synchronous lines (Switched/Non-switched)***
  - ***Hayes AutoSync protocol support***
  - ***V25.bis Autodial and Autoanswer***

IBM Operating System/2™

SNA Phone Connection Support

- *Asynchronous Connectivity Support*
  - *Workstation with COM port or Adapters configured as serial port and Modems 100% compatible with Hayes AT command set*

- *Synchronous connectivity support*
  - *bit oriented V.25bis Autodial and Autoanswer for synchronous modems that implement V.25bis*
  - *Synchronous adapters*
    - *IBM MultiProtocol Communications Adapter for ISA bus*
    - *IBM PS/2 MultiProtocol Adapter/A for Micro Channel*
    - *IBM WAC for Micro Channel and ISA bus*

- *Hayes AutoSync (SDLC and X.25)*
  - *Workstation with COM port or Adapters configured as serial port and Hayes Compatible AutoSync Modem*

IBM Operating System/2™

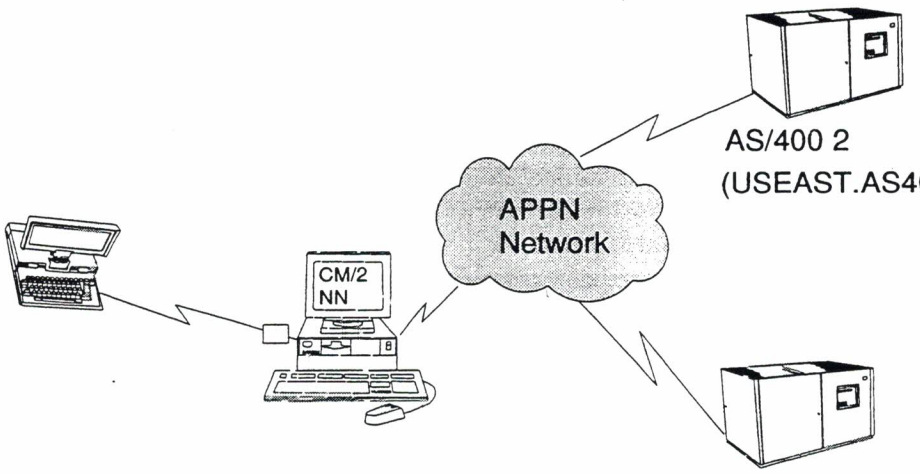
5250 Configuration using SNA Phone Connection

- **OBJECTIVE**
  - **Review quick configuration windows**
  - **Configure 2 sessions, each to a different AS/400**

- **PROCEDURE**
  - **Type CMSETUP or Select CM Setup ICON**
  - **Select 'Setup' pushbutton from CMSETUP**
  - **Choose configuration name**
  - **Select** *5250 Emulation using SNA Phone Connections*

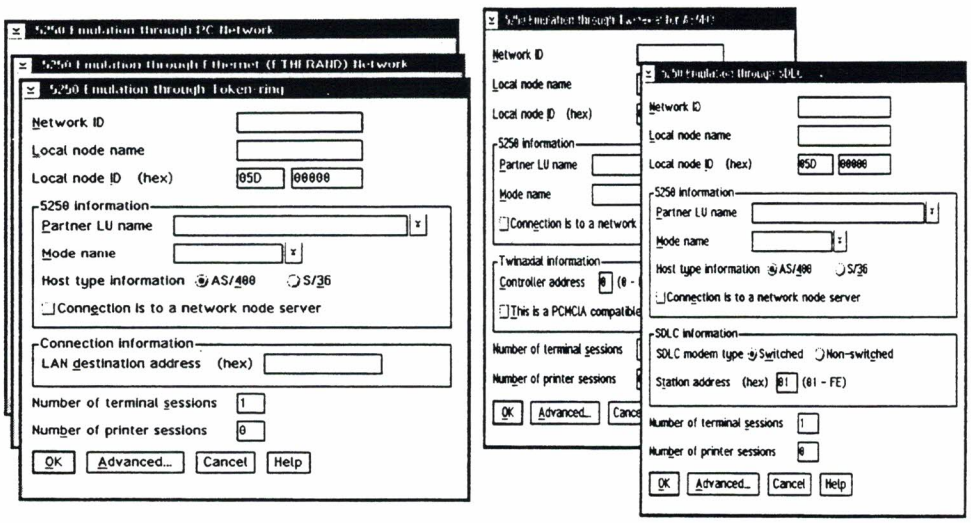
IBM Operating System/2™

5250 Network View



IBM Operating System/2 ™

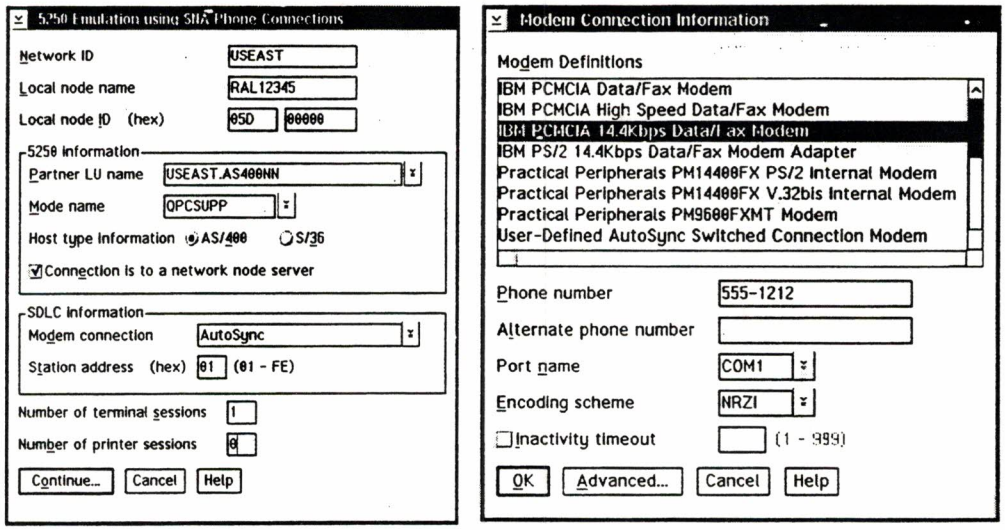
5250 Quick Configuration



Fill in values, select 'Advanced'

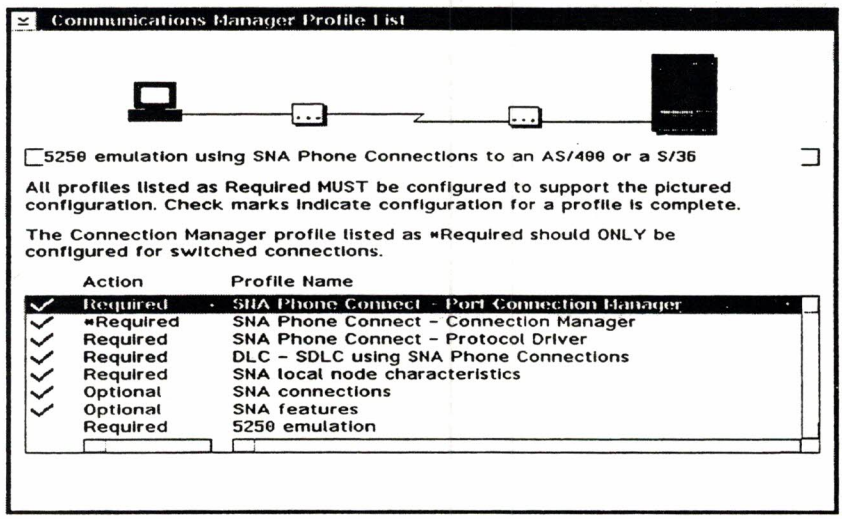
IBM Operating System/2 ™

5250 SNA Phone Connect Quick Configuration



IBM Operating System/2

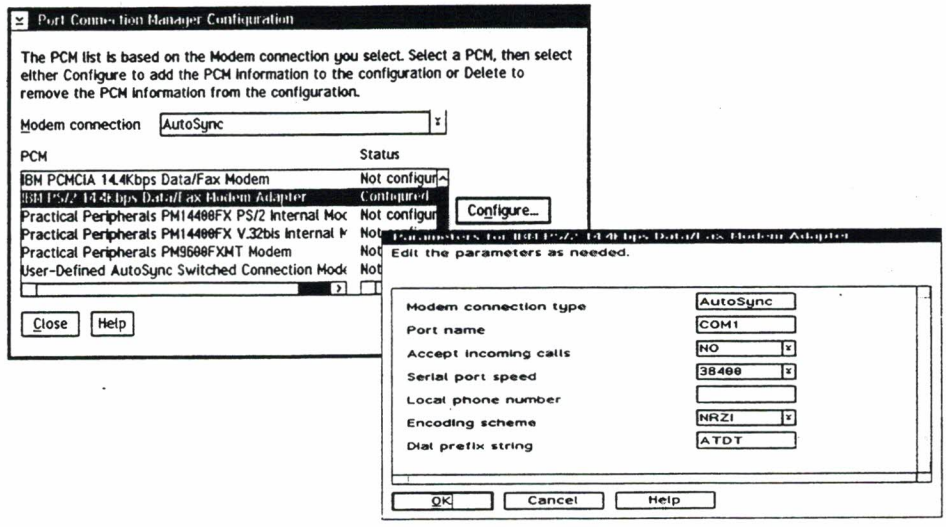
5250 Profile List for SNA Phone Connect



Select profile, then 'Configure...'

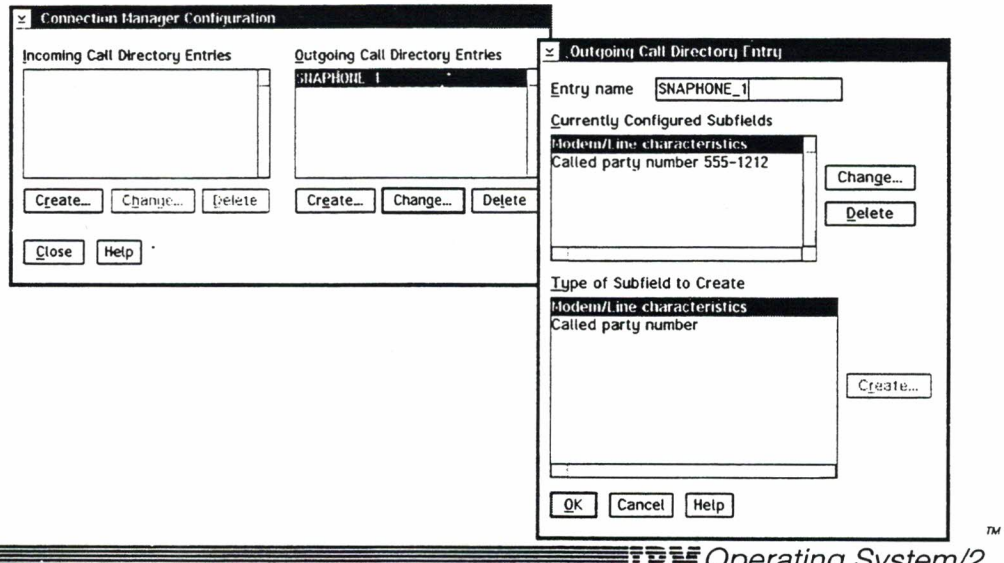
IBM Operating System/2

Port Connection Manager



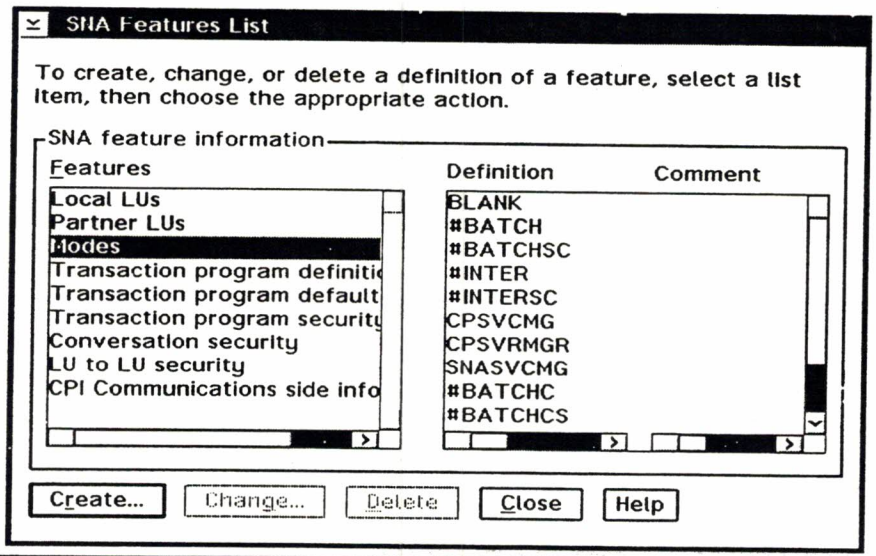
IBM Operating System/2™

Connection Manager



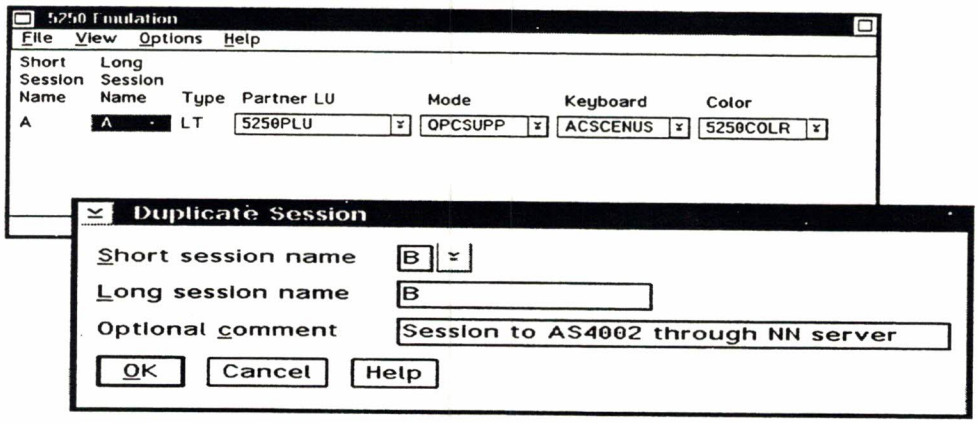
IBM Operating System/2™

SNA Features



IBM Operating System/2™

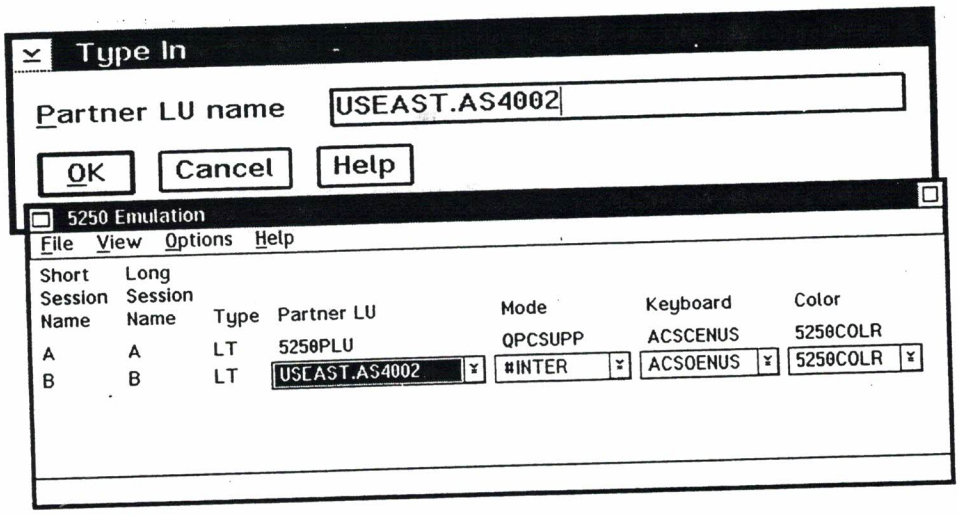
5250 Emulation



Select 'File', Duplicate.

IBM Operating System/2™

5250 Emulation...



IBM Operating System/2™

Agenda

- Advanced Configuration Overview
- 3270 Configuration to multiple hosts
- Gateway Configuration
- 5250 Configuration *including async dialup*
- APPC/APPN Configuration
- CID Support
- Dependent LU Requester

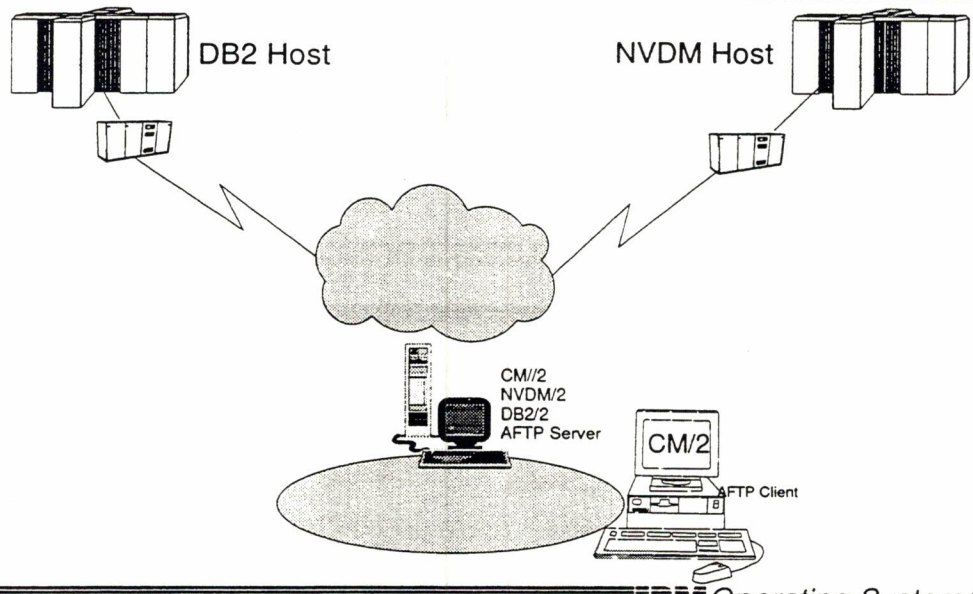
IBM Operating System/2™

APPC/APPN Configuration

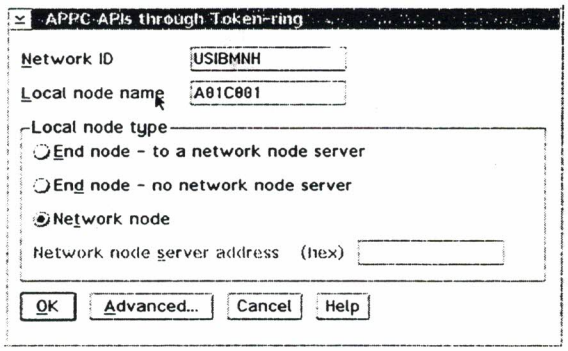
- OBJECTIVE
  - Configure APPC Application AFTP
  - Configure connection to DB2 Host
  - Configure connection to NVDM Host

- PROCEDURE
  - Type CMSETUP or select CM Setup ICON
  - Select 'Setup' pushbutton from CMSETUP
  - Choose configuration name
  - Select 'APPC APIs through Token Ring'
  - Select 'Advanced' pushbutton

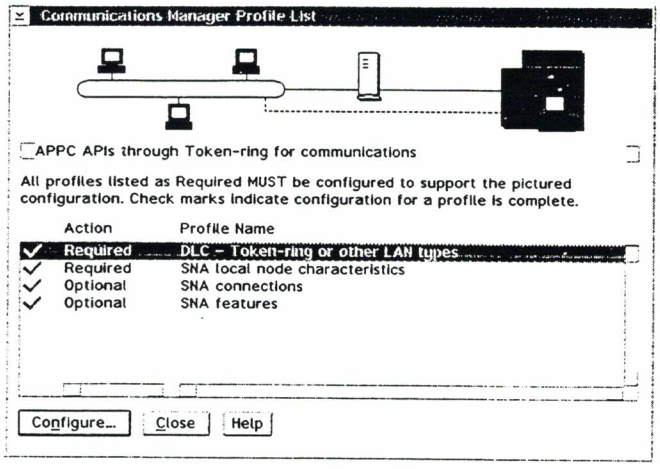
Network View



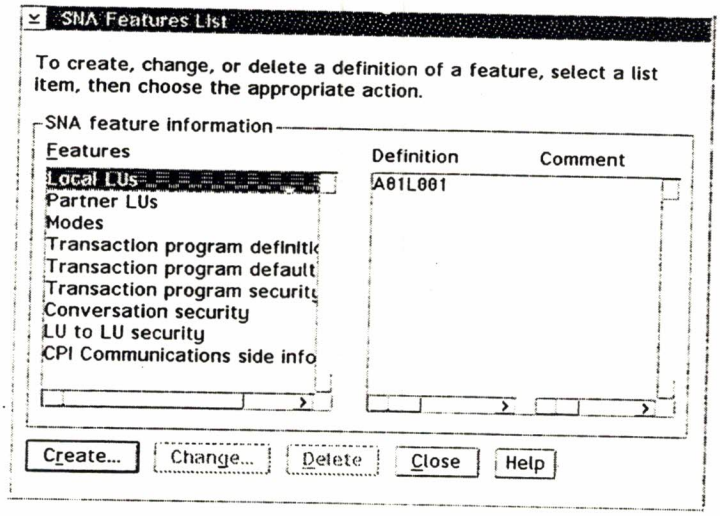
APPC Quick Configuration



APPC Profile List

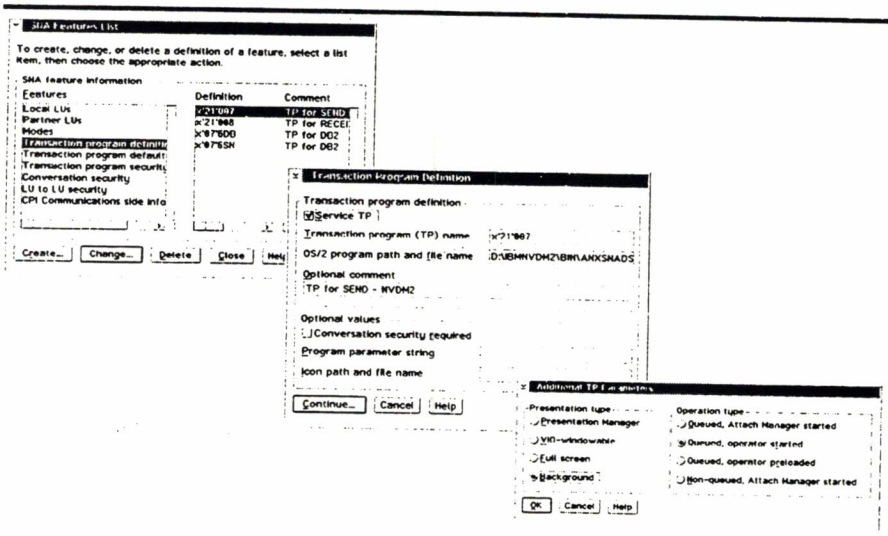


SNA Features



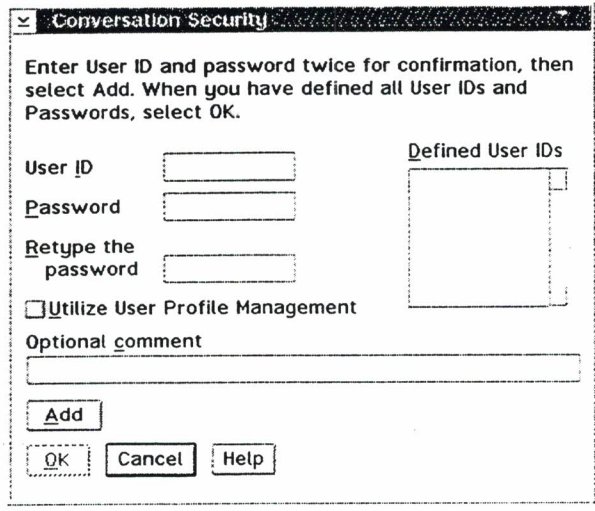
IBM Operating System/2™

Define TP

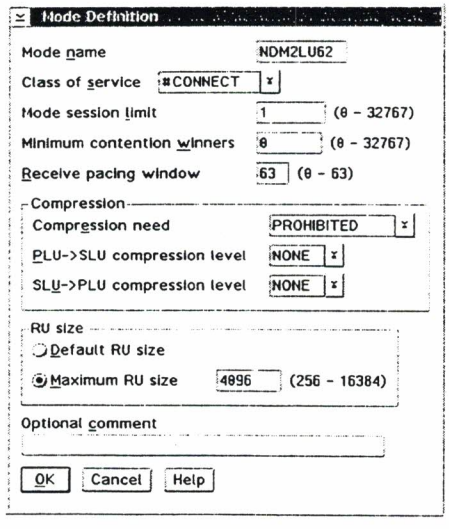


IBM Operating System/2™

Conversation Security (AFTP)



Define Mode (NVDM)



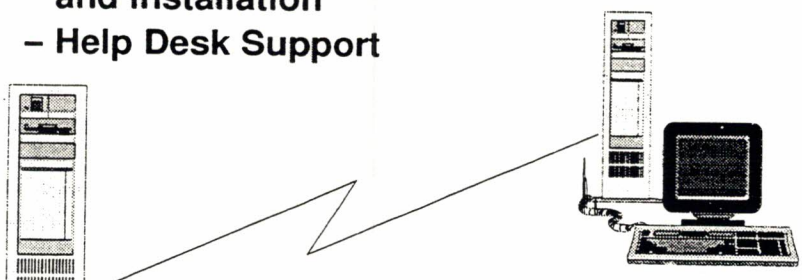
Agenda

- Advanced Configuration Overview
- 3270 Configuration to multiple hosts
- Gateway Configuration
- 5250 Configuration *including async dialup*
- APPC/APPN Configuration
- CID Support
- Dependent LU Requester

IBM Operating System/2™

CID Overview

- Configuration, Installation, and Distribution
- Unattended installation and configuration of OS/2 software
  - Mass installation/configuration
  - Application querying/updating configuration and installation
  - Help Desk Support



IBM Operating System/2™

Communications Manager/2 CID Commands

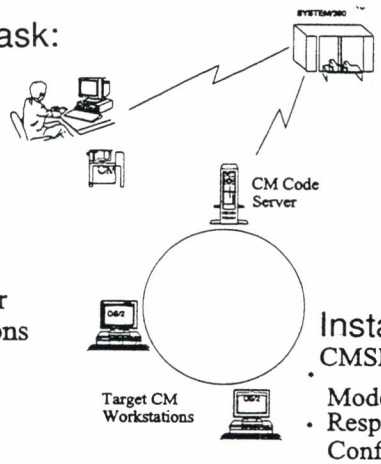
- CMIMAGE - copies diskette images to hard drive
- CMRECORD - builds response file from existing configuration
- CMSETUP /R - install/configure using response file

IBM Operating System/2™

CID Scenario

Administration Task:

- Run CMIMAGE to copy CM diskette files to Hard Drive
- Create Model Configuration and Exception Response Files for Target Workstations



Distribution Task:

- Send files from Administrator's Workstation to CM Code Server

Installation Task:

- CMSETUP /R Command Entered
- Model Configuration & Exception Response Files Processed, User Configuration File Created
- Needed CM Code Installed from CM Code Server

IBM Operating System/2™

CM/2 CID Tips and Techniques

- **Response file migration (/MG parameter)**
- **New response file (/CR parameter)**
- **Supported connectivities**
- **Using a model configuration**
- **Sample response files (UPGRADE.RSP, *REINSTL.RSP*)**
- ***CMUPDATETYPE = 1 or 2 (works the same)***
- **Copying and deleting profiles**
- **Working with keyboard profiles**
- **Using CMRECORD output**

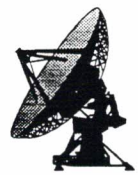
Response file example

```
* Installation values
CMUpdateType = 1
CMWorkstationType = 1
CMTarget = D:
CMModelCFG = X:\MODEL
CMUserCFG = D:\CMCONFIG

CMInstallROPS = 1
CMInstallKeylock = 0
CMUserExit = X:\CMEXIT2 parm
```

```
* Configuration values
Enhanced_keyboard = (
  name = group2kb
  copy = modelkb
  source_cfg = X:\model
)

Enhanced_keyboard = (
  name = oldkb
  delete
)
```

Agenda

- Advanced Configuration Overview
- 3270 Configuration to multiple hosts
- Gateway Configuration
- 5250 Configuration *including async dialup*
- APPC/APPN Configuration
- CID Support
- ► Dependent LU Requester



IBM Operating System/2™

DLUR Configuration

- ► OBJECTIVE
  - Describe process to access CM/2 1.11 DLUR
  - Describe configuration of CM/2 Gateway using DLUR
- ► PROCEDURE
  - Requires VTAM V4R2
  - Edit .NDF file, configure DEFINE_DEPENDENT_LU_SERVER, configure link via CMSETUP

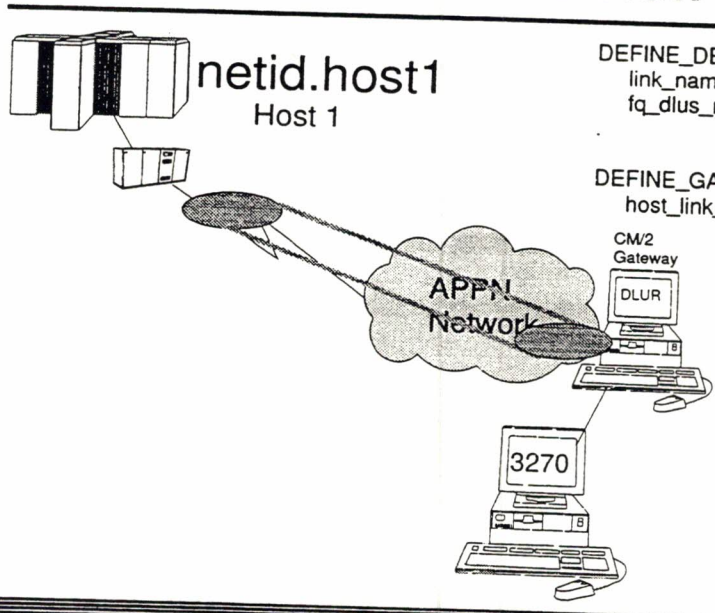
IBM Operating System/2™

DLUR Overview

- Benefits:
  - Support for SSCP ownership is provided to dependent LUs that are either adjacent or non-adjacent to the VTAM or NCP boundary function.
  - APPN dynamics enhanced
- Limitations:
  - Does not support XRF & XRF/CRYPTO, Takeover/Giveback, 3270 Gateway DSPUs, SDDLU
- Available:
  - VTAM V4R2 SYS1.SISTDAT1 library
- Publications:
  - Technical Overview: GG66-3256-00
  - VTAM Network Implementation SC31-6494

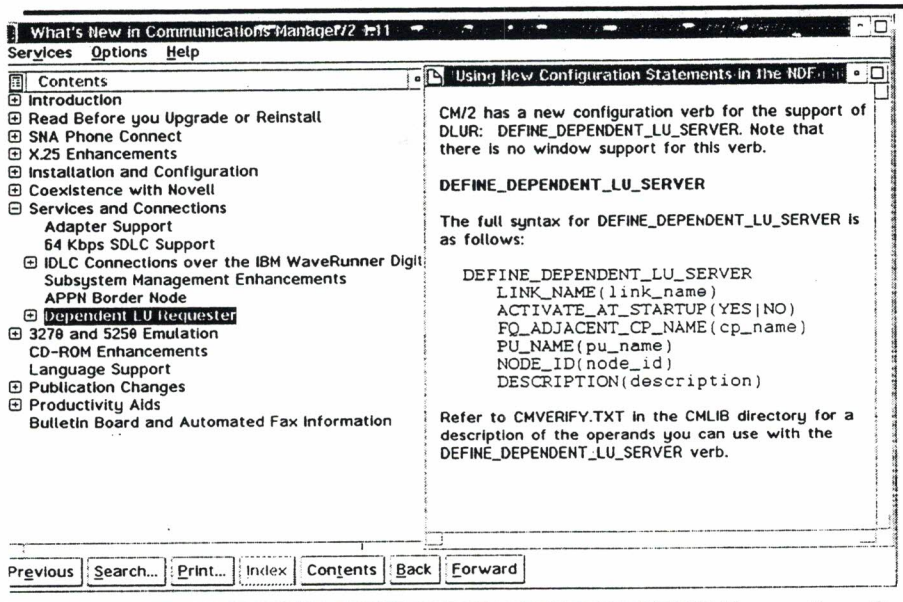
IBM Operating System/2™

DLUS/DLUR Network View



IBM Operating System/2™

DLUR Configuration



Summary

- Guided path for advanced configuration
- Use of profile lists
- Common configuration interfaces
- CID support for volume installations
- Go for it!

Configuration Utilities

- Keylock Configuration (KEYLOCK.EXE)
- Replace Default Configuration (CMREPL.EXE)
- Verify Configuration (CMVERIFY.EXE)
- Upgrade Configuration (CMUPGRAD.EXE)
- Problem Determination
  - PMDSPLAY.EXE, DISPLAY.EXE
  - QSYSTEM.EXE
  - CMQUERY.EXE
  - CHK4DLLS.EXE (Productivity Aid)
- Copy Configuration
  - CMBKCFG.CMD
  - CMRSTCFG.CMD
    - on Network Administrator Sample Program diskette

IBM Operating System/2™

Connection/Feature Combinations

WKST CONNECTION TYPE VS FEATURE OR APPLICATION								
Connection	3270	5250	ACDI Redir	APPC APIs	CPI-C APIs	LUA APIs	ACDI APIs	X.25 APIs
Token-ring or other LAN types	Q	Q	✓	Q	✓	✓		
Ethernet	Q	Q	✓	Q	✓	✓		
PC Network	Q	Q	✓	Q	✓	✓		
Coaxial (DFT)	Q							
Twinaxial	✓	Q		✓	✓	✓		
Asynchronous							✓	
SDLC	Q	Q		✓	✓	✓		
X.25	✓	✓		✓	✓	✓		✓
SDLC using SNA Phone	Q	Q		✓	✓	✓		
X.25 using SNA Phone	✓	✓		✓	✓	✓		✓
IDLC using SNA Phone	✓	✓		✓	✓	✓		

Q Indicates that quick configuration is available
✓ Indicates that this combination is available

IBM Operating System/2™

Configuration Files

Communications Manager Configuration Files					
File	Type	Description	EE	ES	CM
\CMLIB\configname.CFG	Binary	CM configuration file containing Non-SNA data	X	X	X
\CMLIB\configname.NDF	ASCII	CM node definitions file containing SNA data		X	X
\CMLIB\configname.CF2	Binary	CM SNA configuration file		X	X
\CMLIB\configname.SEC	Binary	CM security file		X	X
\IBMCOM\PROTOCOL.INI	ASCII	NTS/2 (LAPS) and SNA Phone configuration file		X	X
\CMLIB\NETWORK.INI	ASCII	Port connection manager configuration file			X
\CMLIB\switchtype.CXM OR \CMLIB\configname.CXM	Binary	Connection manager configuration file			X

IBM Operating System/2™

PRO 1003

OS/2 Workplace Shell (Accept No Substitutes)

Evan Bergman

Development Technology, Inc
Forest Acres, So. Carolina

No paper available. Please contact speaker

this page intentionally left blank

PRO 1005
Object Persistence: Beyond Object-Oriented Databases

Roger Sessions

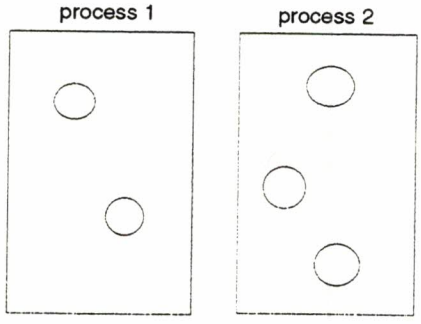
Notices

- This presentation describes work in progress. The products described are currently intended to ship with SOM 3.0, however interfaces and plans may changed without notice.
- This entire presentation is copyright by ObjectWatch, Inc., Austin, Texas. It may not be redistributed, in whole or in part, without permission.

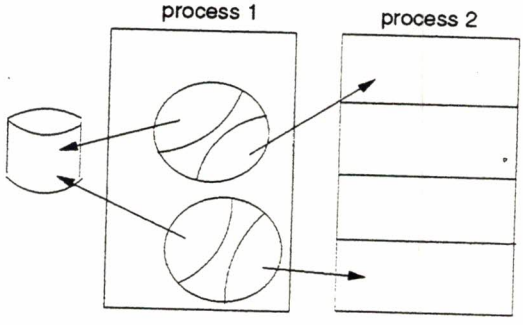
Overview of Talk

- The Evolution of Objects
- Persistence Background
- The Client View
- The Object Implementor View
- The Datastore Provider View
- Unresolved Issues

Objects in the 80's



Objects in the 90's



Objects in the 2000's

Characteristics of Objects in 2000's

- Highly Cooperative
- Highly Interactive, not only with humans, but other objects
- Highly adaptable
- Still integrated in Company Decision Support Systems

Characteristics of Applications in the future

- A few thousand lines of code manipulating vast collections of objects
- each elegantly designed for specific purposes,
- cooperating with each other in a highly networked interdependent manner,
- with new relationships being formed constantly,
- making information available in a truly egalitarian manner.

The Distributed Objects Paradigm

- Objects are probably not in your address space
- Objects were written by third parties
- Objects exist before you got there
- Objects exist after you leave
- Objects are very large
- To conceptualize, think of the World Wide Web

Where is this driving us?

- Object should be strictly business oriented
- Objects should be simple
- Non business code needs another home
  - Database Access Code
  - Security Code
  - Distribution Code
  - Etc.

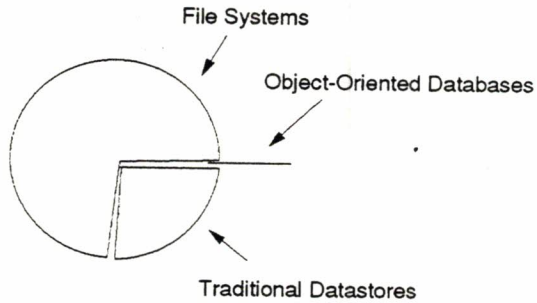
OMG is providing the answer

- Object Management Group
- Over 500 members
- Developing standards for Distributed Objects
  - Life cycle
  - Events
  - Persistence
- Purpose is to enable a Distributable Components Industry

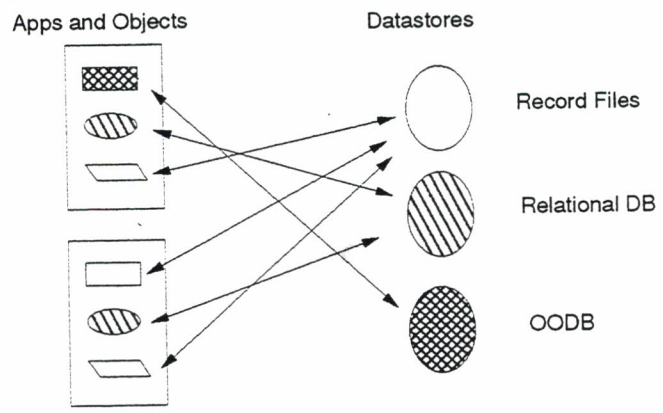
Object Persistence

- Why not use Object-Oriented Databases?
  - Optimized for objects
  - The wave of the future
  - Standards now available
  - "...storing data for object applications in rows and columns makes about as much sense as taking apart a 747 before putting it in the hangar." (From advertisement for major OODB vendor)

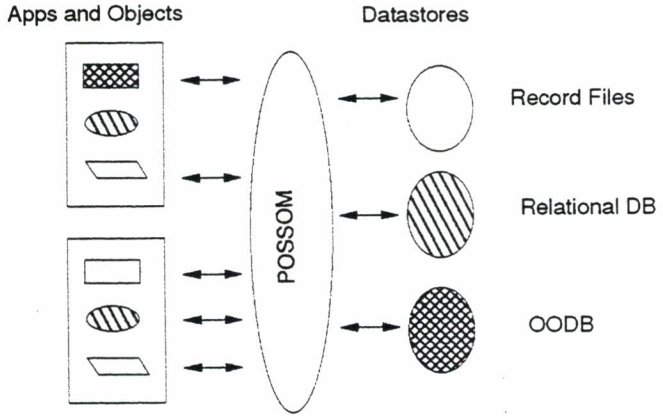
Relative Importance of Datastores



The World Faced by Applications Developers



The Goal of POSSOM



Client View

- General Life Cycle Issues
- The POSSOM Client Level Interfaces
- Client Code

General Life Cycle Issues

- Factories
  - A factory knows how to create objects of some type
  - A given factory will create objects in one process
- FactoryFinder
  - Knows how to locate Factories
  - One per address space
- General use algorithm:
  - factoryFinder = getFactoryFinder();
  - dogFactory = findFactory(factoryFinder, "dog");
  - lassie = _create(dogFactory, "Lassie");

The POSSOM Client Level Interfaces

- PID
  - An object that contains location information
  - Clients use some datastore specialization of PID
- PIDFactory
  - An object that knows how to create a certain type of PID
- PO
  - An interface that allows client control of when objects are stored/restored
- POFactory
  - An object that knows how to create a certain type of PO

Sample Class

```
interface company : CosPersistenceLPO::ImplicitPO {

 const long MaxPeople = 20;
 void add_person(in person newPerson);
 void print_company(in void *output);

 implementation {
  sequence <person, MaxPeople> people;
  override: somInit, somUninit, externalize_to_stream,
        internalize_from_stream;
 };
};
```

The POSSOM Client Interfaces (continued)

- ImplicitPO
  - An interface that allows the Transaction Service to control store/restore
- ImplicitPOFactory
  - An object that knows how to instantiate an ImplicitPO object

Client Code 1/4

```
main ()
{
/* Set up.
   ------- */
   company ABC;
   Environment *ev;

   CosPersistencePOSIX_PID_POSIX pidCompany;
   CosPersistencePOM_POM pom;
   CosPersistenceTrans_TransMgr transMgr;

   CosPersistenceIPO_ImplicitPOFactory companyFact;

   CosPersistencePID_PIDFactory pidf;
   CosPersistencePOF_FactoryFinder ff;
```

Client Code 2/4

```
/* Get global objects.
   ------------------- */
   ev = SOM_CreateLocalEnvironment();
   ff = getGlobalFactoryFinder(ev);
   transMgr = getGlobalTransMgr(ev);

/* Instantiate factories.
   ---------------------- */
   pidf = _findFactory (ff, ev, "CosPersistencePOSIX::PID_POSIX",
          CosPersistencePOF_type);
   companyFact = _findFactory (ff, ev, "company",
          CosPersistencePOF_type);
```

Client Code 3/4

```
/* Prepare Company pid.
   ------------------- */
   pidCompany = _create(pidf, ev);
   __set_datastore_type(pidCompany, ev, "File1");
   __set_pathName(pidCompany, ev, "comp.d19");

/* Create company.
   -------------- */
   ABC = _createImplicitPO(companyFact, ev, pidCompany);
```

Client Code 4/4

```
/* Print company.
   ----------------- */
   _startTrans(transMgr, ev);
  _print_company(ABC, ev, output);
  commitTrans(transMgr, ev);

}
```

Object Implementor (OI) View

- The OI Interfaces
- Deriving Object from ImplicitPO
- Overriding the Streamable Methods

The OI Interfaces - ImplicitPO

```
module CosPersistenceIPO
 {
   interface ImplicitPO : CosPersistencePO::PO {
   };
};
```

The OI Interfaces - StreamIO

```
module CosStream
{
  interface StreamIO : SOMObject
   {
     short read_short();
     void write_short(in short item);

     long read_long();
     void write_long(in long item);
/*   ... */
     Streamable read_object();
     void write_object(in Streamable item);
   };
```

The OI Interfaces - Stream

```
interface Streamable : SOMObject
{
  void externalize_to_stream(in CosStream::StreamIO stream);
  void internalize_from_stream(in CosStream::StreamIO stream);
};
```

Deriving Object from ImplicitPO

```
interface company : CosPersistencePO::ImplicitPO {

  const long MaxPeople = 20;
  void add_person(in person newPerson);
  void print_company(in void *output);

  implementation {
    sequence <person, MaxPeople> people;
    override: somInit, somUninit, externalize_to_stream,
          internalize_from_stream;
   };
};
```

Override externalize_to_stream

```
static void externalize_to_stream(
company somSelf,
Environment *ev, CosStream_StreamIO stream)
{
  long n;
  person next;
  companyData *somThis = companyGetData(somSelf);

  _write_long(stream, ev, _people._length);
  for (n=0; n< _people._length; n++) {
     next = _people._buffer[n];
     _write_object(stream, ev, next);
  }
}
```

Override internalize_from_stream

```
static void internalize_from_stream(1company somSelf,
  Environment *ev,CosStream_StreamIO stream)
{
  long n, npeople;
  person next;
  companyData *somThis = companyGetData(somSelf);
  npeople = _read_long(stream, ev);
  for (n=0; n<npeople; n++) {
    next = _read_object(stream, ev, NULL, NULL);
   _add_person(somSelf, ev, next);
  }
}
```

DataStore Provider View

- Any datastore can be an object-oriented database!
  - Derive a new Persistent ID
  - Specialize a Persistent Data Service from (PDS)
  - Define how to do mappings from objects (optional)

Unresolved Issues

- How will object security relate to datastore security?
- How will we "find" objects?
- What will "persistent" collections look like?

Summary

- OMG is developing standards for Distributed Objects
- The Persistent Object Service (POS) defines
  - How objects are made persistent.
  - How databases are made into object data storage facilities.
  - How clients manipulate persistent objects.
- POS Advantages
  - Distributed Objects can be stored...
  - using Industry Standard technology...
  - in the databases you already know and love!

About the Author

Roger Sessions has spoken at dozens of conferences throughout the world and has written extensively about SOM and Object-Oriented Programming. He has given SOM tutorials at OOPSLA and other conferences and has consulted on many large scale SOM projects.

Roger was part of the IBM SOM development team from 1990-1995. He was the technical lead for the SOM Persistence Framework and a principal architect of the OMG's Persistent Object Service, an industry standard for Object Persistence based on the SOM Persistence Framework

In 1995, Roger left IBM to form ObjectWatch, Inc., a company specializing in consulting, training, and mentoring in the use of SOM, DSOM, and other aspects of Distributed Object Technology.

Roger is the author of three books. His most recent book, scheduled for release in late 1995, is *Object Persistence: Beyond Object-Oriented Databases* , an upfront and personal view of the OMG Persistence Specification. He has also written *Class Construction in C and C++; Object-Oriented Fundamentals* and *Reusable Data Structures for C*. Roger has co-authored many articles in OS/2 publications.

He can be reached at roger@fc.net.

PRO 1006
Making Concurrent Messages under OS/2 2.X

Luca Miccoli
IBM
Italy

Dr. Luca Miccoli
IBM SEMEA S.p.A., Rome Networking Software Laboratory
P.le G.Pastore 6, 00144 Rome - ITALY
Internet: lmiccoli@vnet.ibm.com

ABSTRACT:

This paper is the continuation of Making Concurrent Messages Under OS/2 2.x published in the OS/2 World Conference & Exhibition Proceedings (Volume 2), July 20-22, 1994 Santa Clara, California (Miller Freeman, Inc. - ISBN # 0-87930- 347-6). The value added to the mentioned paper consists of a C++ code example based on the IBM User Interface Class Libraries (ICLUI).

KEY WORDS:

OS/2 Details View Containers; Pop-up menus; Threads; Event and Mutex semaphores; C++ language used in the example.

1. C++ CODE EXAMPLE

The C++ code example contained in the following paragraphs is composed by the following files:

MSGVWR.HPP	include file with all the Message Viewer window classes declarations
MSGVWR.CPP	implementation of the classes that can be used for creating a Message Viewer window and for managing messages inside it (insertion, deletion)
MSGVWR.RC	resource file with the context menus for the Message Viewer window
MSGVWR.RCH	definitions for the IDs of the context menus
TESTMVWR.CPP	sample program that creates a Message Viewer window (using the classes defined in
MSGVWR.HPP)	and inserts some sample messages inside it

As already mentioned in the Abstract, the example code exploits the IBM User Interface Class Libraries (ICLUI) that come with the IBM C Set++ compiler: the assumption is made that the PM application exploiting the Message Viewer classes must be multithreaded. For the didactic purposes of this paper, the code listing of the following paragraphs contains as many comments as possible, to fully highlight all the important logical steps. In order to limiting the number of pages, not much care has been taken to test the validity of the OS/2 APIs return codes and other possible strange error situations (like C++ exceptions thrown by ICLUI); however, these tests are obviously mandatory for quality code, and are really straightforward to implement. Moreover, still for "sizing" reasons, the code example contains some hard-coded character strings, and this must be avoided when writing NLS (National Language Support) enabled code.

2. MSGVWR.HPP LISTING

```
#ifndef MSGVWR_HPP
#define MSGVWR_HPP

// ************************************************************************ //
//
// Module Name                    :  MSGVWR.HPP
//
//  Descriptive Name:  Header file for Message Viewer classes
//
// ************************************************************************ //

#define INCL_DOSSEMAPHORES                                  // Semaphores APIs and
#include <os2.h>                                            // data definition

#include <iframe.hpp>           // IFrameWindow
#include <icmdhdr.hpp>          // ICommandEvent & ICommandHandler
#include <icnrobj.hpp>          // IContainerObject
#include <icnrctl.hpp>          // IContainerControl
#include <icnrcol.hpp>          // IContainerColumn
#include <icnrhdr.hpp>          // ICnrHandler
#include <icnrmhdr.hpp>         // ICnrMenuHandler
#include <ireslock.hpp>         // IPrivateResource

// Forward declarations
class MsgVwrWindow;
class IPointerHandle;

//************************************************************************
// Class:  MsgVwrMessage
//
// Purpose: This class represents a generic message to be inserted in the
//          Message Viewer window
//
//************************************************************************

#pragma pack(1) // 1 byte alignment is mandatory for IContainerObject derived
                // classes
class MsgVwrMessage : public IContainerObject
{
 friend class MsgVwrWindow; // The MsgVwrWindow class is made a friend to the
                            // MsgVwrMessage class so that the IContainerColumn
                            // objects in the IContainerControl object associated
                            // with the Message Viewer window can be initialized
                            // using the "offsetof" macro
 public:

 // Constructor
 MsgVwrMessage(unsigned long ulMsgNumber,                   // Message number
               IString const & MsgText,                     // IString that contains the
                                                            // message text I
               PointerHandle const & MsgIcon,               // Message icon pointer
               MsgVwrWindow * pMsgVwrWin,                   // Pointer to a MsgVwrWindow
                                                            // object
               Boolean fModal=false);                       // Modality flag. If it is set
                                                            // to true, the addToMsgVwr
                                                            // method does not return
                                                            // unless the user has
                                                            // specified a response in the
                                                            // Message Viewer window
```

```
// Add an object of this class in the Message Viewer window
//unsigned long addToMsgVwr();

// Returns the modality attribute
// Boolean isModal() const;

// Sets the processed status with a user response ID for a modal message
void setProcessed(unsigned long ulActionID);

// Returns the processed status for a modal message
Boolean isProcessed() const;

private:
// Suspend the thread execution for a modal message
void lock();

// Resume the thread execution for a modal message
void unlock();

unsigned long  ivulMsgNumber;          // Message number
IString        ivMsgText;              // Message text string
IPointerHandle ivMsgIcon;              // Message icon pointer
char  ivMsgDate[4];                    // Message date
char  ivMsgTime[4];                    // Message time
MsgVwrWindow*  ivpMsgVwrWin;           // Pointer to a MsgVwrWindow object
Boolean        ivfModal;               // Flag that must be set to true or
                                       // to false depending whether the message
                                       // is modal or not
HEV            ivEventSem;             // Handle of the event semaphore that is
                                       // posted when a modal message is processed
unsigned long  ivulActionID;           // User response ID for a modal message
Boolean        ivfProcessed;           // Flag that must be set to true when the
                                       // message received a response

};
#pragma pack()

//*****************************************************************************
// Class:  MsgVwrMenuHandler
//
// Purpose: Subclass of ICnrMenuHandler needed to display the Message Viewer
//          window main and object context (popup) menus.
//
//*****************************************************************************
class MsgVwrMenuHandler: public ICnrMenuHandler
{
public:

// Initialize the IContainerControl * attribute
void setCnr(IContainerControl * pCnr);

protected:
// Override ICnrMenuHandler::makePopUpMenu
Boolean makePopUpMenu(IMenuEvent& cnEvt);

private:
IContainerControl * ivpMsgVwrCnr;  // Pointer to the IContainerControl
                                   // object whose menus must be handled
};
```

```
//*****************************************************************************
// Class:  MsgVwrCmdHandler
//
// Purpose: Subclass of ICommandHandler needed to handle the commands
//          originated by the Message Viewer window context menus
//
//*****************************************************************************
class MsgVwrCmdHandler: public ICommandHandler
{
 public:

 // Initialize the IContainerControl * attribute
 void setCnr(IContainerControl * pCnr);

 protected:
 // Override ICommandHandler::command method
 Boolean command(ICommandEvent& cmdEvent);

 private:
IContainerControl * ivpMsgVwrCnr; // Pointer to the IContainerControl
                                  // object associated with the commands
                                  // to be handled
};

//*****************************************************************************
// Class:  MsgVwrWindow
//
// Purpose: Message Viewer window class
//
//*****************************************************************************
class MsgVwrWindow : public IFrameWindow,
                     public ICnrHandler
{
 public:

 // Constructor
 MsgVwrWindow(unsigned long windowId,              // Message Viewer window ID
              unsigned long ulMaxNumMsg=50); // Maximum number of messages
                                             // in the Message Viewer window
 // Destructor
 MsgVwrWindow ();

 // Insert a message in the Message Viewer window
 void addMessage(const MsgVwrMessage * pMessage);

 private:

 IContainerControl * ivpMsgVwrCnr;          // Pointer to a IContainerControl
                                            // object implementing the Message
                                            // Viewer window container
 IContainerColumn   * ivpColMsgNumber,      // Message number column
                    * ivpColMsgText,        // Message text column
                    * ivpColMsgIcon,        // Message icon column
                    * ivpColMsgDate,        // Message date column
                    * ivpColMsgTime;        // Message time column
 MsgVwrCmdHandler   ivCmdHdr;               // Message Viewer window command
                                            // handler
 MsgVwrMenuHandler  ivMenuHdr;              // Message Viewer window menu handler
 unsigned long  ivulMaxNumMsg;              // Maximum number of messages
                                            // in the Message Viewer window
 IPrivateResource ivMutexSem;               // Private resource needed to serialize
                                            // the insertion of messages
};
```

```
// Inlined class members
inline Boolean MsgVwrMessage::isModal() const
{
 return ivfModal;
}

inline Boolean MsgVwrMessage::isProcessed() const
{
 return ivfProcessed;
}

inline void MsgVwrMenuHandler::setCnr(IContainerControl * pCnr)
{
 ivpMsgVwrCnr = pCnr;
}

inline void MsgVwrCmdHandler::setCnr(IContainerControl * pCnr)
{
 ivpMsgVwrCnr = pCnr;
}
#endif
```

3. MSGVWR.CPP LISTING

```
// ************************************************************************ //
//
// Module Name    :  MSGVWR.CPP
//
// Descriptive Name:  Implementation of Message Viewer classes
//
// ************************************************************************ //

#include <string.h>             // memcpy()
#include <idate.hpp>            // IDate
#include <itime.hpp>            // ITime
#include <ipopmenu.hpp>         // IPopupMenu
#include <icnrolst.hpp>         // ICnrObjectSet
#include <ihandle.hpp>          // IPointerHandle
#include <msgvwr.rch>           // Context menus IDs

#include <msgvwr.hpp>           // Headers of the Message Viewer classes

//************************************************************************
// MsgVwrMessage - Constructor
//************************************************************************
MsgVwrMessage::MsgVwrMessage(unsigned long ulMsgNumber,
                             IString const & MsgText,
                             IPointerHandle const & MsgIcon,
                             MsgVwrWindow * pMsgVwrWin,
                             Boolean fModal)
                            :
                            IContainerObject(),       // Call IContainerObject Constructor
                            ivulMsgNumber(ulMsgNumber),
                            ivMsgText(MsgText),
                            ivMsgIcon(MsgIcon),
                            ivpMsgVwrWin(pMsgVwrWin),
                            ivfModal(fModal),
                            ivulActionID(0),
                            ivEventSem(0)
{ // Start of MsgVwrMessage::MsgVwrMessage
 // Add the "\n...Waiting response..." string to the message text if the
 // message is modal
 if(fModal)
   ivMsgText += "\n...Waiting response...";
} // End of MsgVwrMessage::MsgVwrMessage
```

```
//*****************************************************************************
// MsgVwrMessage - addToMsgVwr
//*****************************************************************************
unsigned long MsgVwrMessage::addToMsgVwr()
{ // Start of MsgVwrMessage::addToMsgVwr
 // Set the current date and time for the message to be inserted
 ivMsgDate[0] = (char)(IDate::today().dayOfMonth());
 ivMsgDate[1] = (char)(IDate::today().monthOfYear());
 unsigned short *pusDate = (unsigned short *)&(ivMsgDate[2]);
 *pusDate = (unsigned short)(IDate::today().year());
 ivMsgTime[0] = (char)(ITime::now().hours());
 ivMsgTime[1] = (char)(ITime::now().minutes());
 ivMsgTime[2] = (char)(ITime::now().seconds());
 ivMsgTime[3] = '\0';

 // Add the message in the Message Viewer window
 ivpMsgVwrWin->addMessage(this);

// Wait for a user response if the message is modal
 if(this->isModal()) {
   this->lock();
   DosCloseEventSem(ivEventSem);
 }
 else
   ivfProcessed = true;

 // Return the user response
 return ivulActionID;
} // End of MsgVwrMessage::addToMsgVwr

//*****************************************************************************
// MsgVwrMessage - lock
//*****************************************************************************
void MsgVwrMessage::lock()
{ // Start of MsgVwrMessage::lock
 // Create an event semaphore and wait until it is posted (this is actually
 // done after the user provides a response to the message in the Message
 // Viewer window)
 DosCreateEventSem((char *)0, &ivEventSem, 0L, 0L);
 DosWaitEventSem(ivEventSem, SEM_INDEFINITE_WAIT);
 return;
} // End of MsgVwrMessage::lock


//*****************************************************************************
// MsgVwrMessage - unlock
//*****************************************************************************
void MsgVwrMessage::unlock()
{ // Start of MsgVwrMessage::unlock
 // Post the event semaphore created in the lock method
 DosPostEventSem(ivEventSem);
 return;
} // End of MsgVwrMessage::unlock
```

```
//*****************************************************************************
// MsgVwrMessage - setProcessed
//*****************************************************************************
void MsgVwrMessage::setProcessed(unsigned long ulActionID)
{ // Start of MsgVwrMessage::setProcessed
 // Set to true the ivfProcessed attribute
 ivfProcessed = true;
 if(ivfModal) { // The message is modal
  // Set the ivulActionID attribute
  ivulActionID = ulActionID;

  // Remove the "\n...Waiting response..." string from the message
  ivMsgText.remove(ivMsgText.lastIndexOf("\n...Waiting response..."));

  // Call the unlock method to let the addToMsgVwr method return
  this->unlock();
 }

 return;
} // End of MsgVwrMessage::setProcessed

//*****************************************************************************
// MsgVwrMenuHandler - makePopUpMenu
//*****************************************************************************
Boolean MsgVwrMenuHandler::makePopUpMenu(IMenuEvent& cnEvt)
{ // Start of MsgVwrMenuHandler::makePopUpMenu
 IPopUpMenu * popUp;              // Pointer to a IPopUpMenu object
 unsigned long ulSelMessages=0; // Number of selected messages in the
                                  // Message Viewer Window
 IContainerControl::ObjectCursor MsgCursor (*ivpMsgVwrCnr); // Cursor for the
                                                            // Message Viewer IContainerControl object

 // Get the message for which a pop-up menu was requested. If a pop-up
 // menu was not requested for a message, 0 is returned.
 MsgVwrMessage * pMessage = (MsgVwrMessage *) popupMenuObject();

 if(pMessage) { // Create a Message Viewer object context menu
  // Set the cursor to the record
  ivpMsgVwrCnr->setCursor(pMessage);

  // Initialize ulSelMessages
  for (MsgCursor.setToFirst(); MsgCursor.isValid(); MsgCursor.setToNext()) {
   if(ivpMsgVwrCnr->isSelected(ivpMsgVwrCnr->objectAt(MsgCursor)))
    ulSelMessages++;
  }

  // Create an IPopUpMenu object
  popUp=new IPopUpMenu(ID_MENU_MSGVWR_OBJ,
                       cnEvt.window());

  // Disable the "Delete", "Message help", "Ok" and "Cancel" menu items
  // if more than 1 message is selected and the record associated with
  // pMessage is selected
  if(ulSelMessages > 1 && ivpMsgVwrCnr->isSelected(pMessage)) {
   popUp->disableItem(IDM_MSGVWR_DEL_ONE);
   popUp->disableItem(IDM_MSGVWR_HELP_FOR_OBJ);
   popUp->disableItem(IDM_MSGVWR_OK);
   popUp->disableItem(IDM_MSGVWR_CANCEL);
  }
```

```
// Disable the "Ok" and "Cancel" menu items if the message has already
// been processed
if(pMessage->isProcessed()) {
 popUp->disableItem(IDM_MSGVWR_OK);
 popUp->disableItem(IDM_MSGVWR_CANCEL);
}

// Disable the "Delete" menu item if the message is modal and not processed
if(pMessage->isModal() && !(pMessage->isProcessed()))
 popUp->disableItem(IDM_MSGVWR_DEL_ONE);

} // End if - Create a Message Viewer object context menu

else { // Create a Message Viewer main context menu
 // Create an IPopUpMenu object
 popUp=new IPopUpMenu(ID_MENU_MSGVWR_MAIN,
                      cnEvt.window());

 // Initialize ulSelMessages
 for (MsgCursor.setToFirst(); MsgCursor.isValid(); MsgCursor.setToNext()) {
  if(lvpMsgVwrCnr->isSelected(lvpMsgVwrCnr->objectAt(MsgCursor)))
    ulSelMessages++;
  }

 // Disable the "Deselect all" and "Delete" menu items
 // if the number of selected messages is = 0.
 if(ulSelMessages == 0) {
   popUp->disableItem(IDM_MSGVWR_DESEL_ALL);
   popUp->disableItem(IDM_MSGVWR_DELETE);
 }

 // Disable the "Select all" menu item if the Message
 // Viewer container does not contain records
 if(lvpMsgVwrCnr->objectCount() == 0)
   popUp->disableItem(IDM_MSGVWR_SEL_ALL);
} // End else - Create a Message Viewer main context menu

// Destroy the context menu after it ceases to be displayed
popUp->setAutoDeleteObject();

// Show the context menu
popUp->show(cnEvt.mousePosition());

return true;
} // End of MsgVwrMenuHandler::makePopUpMenu
```

```
//*****************************************************************************
// MsgVwrCmdHandler - command
//*****************************************************************************
Boolean MsgVwrCmdHandler::command(ICommandEvent& cmdevt)
{ // Start of MsgVwrCmdHandler::command
 switch(cmdevt.commandId()) { // Start of switch


  case IDM_MSGVWR_HELP:
  {
  // Show the help panel for the Message Viewer window
  // ...
  // ...
  // ...
  }
  break;

  case IDM_HELPUSINGHELP:
  {
  // Show the Help for help panel
  // ...
  // ...
  // ...
  }
  break;

  case IDM_HELPKEYS:
  {
  // Show the Keys help panel
  // ...
  // ...
  // ...
  }
  break;

  case IDM_HELPINDEX:
  {
  // Show the Help index panel
  // ...
  // ...
  // ...
  }
  break;

 case IDM_MSGVWR_OK: // The user chose the "OK" menu item for the
 {                   // selected message
  // Query selected record
  MsgVwrMessage * pSelMsg= (MsgVwrMessage *)
                           (this->ivpMsgVwrCnr->cursoredObject());
  // Process the message
  if(pSelMsg->isModal()) {
    pSelMsg->setProcessed(IDM_MSGVWR_OK);
  }
 }
break;
```

```
case IDM_MSGVWR_CANCEL: // The user chose the "Cancel" menu item for
{                                   // the selected message
 // Query selected record
 MsgVwrMessage * pSelMsg= (MsgVwrMessage *)
                                (this->ivpMsgVwrCnr->cursoredObject());
 // Process the message
 if(pSelMsg->isModal()) {
  pSelMsg->setProcessed(IDM_MSGVWR_CANCEL);
 }
}
break;


case IDM_MSGVWR_SEL_ALL:  // Select all messages
{
 IContainerControl::ObjectCursor MsgCursor (*ivpMsgVwrCnr); // Cursor for the
                                        // Message Viewer IContainerControl object
 for (MsgCursor.setToFirst(); MsgCursor.isValid(); MsgCursor.setToNext()) {
  try {
    ivpMsgVwrCnr->setSelected(ivpMsgVwrCnr->objectAt(MsgCursor));
  }
  catch (IException& exc) {} // Catch any exceptions to enable the loop
                                // to continue
}
break;

case IDM_MSGVWR_DESEL_ALL: // Deselect all messages
{
 IContainerControl::ObjectCursor MsgCursor (*ivpMsgVwrCnr); // Cursor for the
                                      // Message Viewer IContainerControl object
 for (MsgCursor.setToFirst(); MsgCursor.isValid(); MsgCursor.setToNext()) {
  try {
   ivpMsgVwrCnr->removeSelected(ivpMsgVwrCnr->objectAt(MsgCursor));
  }
  catch (IException& exc) {} // Catch any exceptions to enable the loop
                                // to continue
}
break;


case IDM_MSGVWR_DEL_ONE:              // Delete 1 selected message
{
 MsgVwrMessage * pSelMsg= (MsgVwrMessage *)
                                (this->ivpMsgVwrCnr->cursoredObject());
 if(pSelMsg->isProcessed())
  ivpMsgVwrCnr->removeObject(pSelMsg);
}
break;
```

```
case IDM_MSGVWR_DELETE:                  // Delete selected messages
{
 IContainerControl::ObjectCursor MsgCursor (*ivpMsgVwrCnr);
 // Cursor for the // Message Viewer IContainerControl object
 ICnrObjectSet MsgsToBeDeleted; // Temporary set that contains all the
                                // messages to be removed
 MsgVwrMessage * pSelMsg=0;          // Pointer to a selected message

 // Loop to add elements in the MsgsToBeDeleted set
 for(MsgCursor.setToFirst(); MsgCursor.isValid(); MsgCursor.setToNext()) {
  pSelMsg = (MsgVwrMessage *) (ivpMsgVwrCnr->objectAt(MsgCursor));
  If(ivpMsgVwrCnr->isSelected(pSelMsg) && pSelMsg->isProcessed()) {
    MsgsToBeDeleted.add(pSelMsg);
  }
 }

 // Remove all the messages in the MsgsToBeDeleted set
 ICnrObjectSet::Cursor MsgDelCursor (MsgsToBeDeleted);
 for(MsgDelCursor.setToFirst(); MsgDelCursor.isValid();
     MsgDelCursor.setToNext())
   ivpMsgVwrCnr->removeObject(MsgDelCursor.element());
}
break;


case IDM_MSGVWR_HELP_FOR_OBJ: // Show help for 1 selected message
{
  MsgVwrMessage * pSelMsg= (MsgVwrMessage *)
                           (this->ivpMsgVwrCnr->cursoredObject());
 // Show help for the selected message
 // ...
 // ...
 // ...
}
break;

default:
 break;

} // End of switch

return(false);
} // End of MsgVwrCmdHandler::command
```

```
//*****************************************************************
// MsgVwrWindow - Constructor
//*****************************************************************
MsgVwrWindow::MsgVwrWindow(unsigned long windowId,
                           unsigned long ulMaxNumMsg)
 : IFrameWindow (                    // Call IFrameWindow constructor
                 "Message Viewer Title",
                 windowId,                    // Message Viewer Window ID
                 IFrameWindow::defaultStyle())     // Use default IFrameWindow style

  ivulMaxNumMsg(ulMaxNumMsg)
{ // Start of MsgVwrWindow::MsgVwrWindow

 // Create a IContainerControl object and initialize the ivpMsgVwrCnr
 // attribute
 ivpMsgVwrCnr=new IContainerControl(ID_CNR_MSGVIEWER, this, this);

 // Attach the ivCmdHdr handler to the Message Viewer window object.
 ivCmdHdr.handleEventsFor(this);

 // Associate ivpMsgVwrCnr with ivCmdHdr
 ivCmdHdr.setCnr(ivpMsgVwrCnr);

 // Attach the ICnrHandler handler to the Message Viewer window object.
 ICnrHandler::handleEventsFor(ivpMsgVwrCnr);

 // Attach the ivMenuHdr handler to the Message Viewer window object.
 ivMenuHdr.handleEventsFor(ivpMsgVwrCnr);

 // Associate ivpMsgVwrCnr with ivMenuHdr
 ivMenuHdr.setCnr(ivpMsgVwrCnr);


 // Allow multiple messages selection
 ivpMsgVwrCnr->setMultipleSelection();

// Create the Message Number column
 ivpColMsgNumber = new IContainerColumn (offsetof(MsgVwrMessage, ivulMsgNumber),
                      IContainerColumn::string | IContainerColumn::alignVerticallyCentered |
                 IContainerColumn::alignCentered | IContainerColumn::readOnlyHeading,
                 IContainerColumn::number | IContainerColumn::alignVerticallyCentered | IContainerColumn::alignLeft |
                IContainerColumn::readOnly |  IContainerColumn::horizontalSeparator);

// Create the Message Text column
 ivpColMsgText = new IContainerColumn (offsetof(MsgVwrMessage, ivMsgText),
                                 IContainerColumn::string | IContainerColumn::alignVerticallyCentered |
                                 IContainerColumn::alignCentered | IContainerColumn::readOnlyHeading,
                                 IContainerColumn::string | IContainerColumn::alignVerticallyCentered |
                                 IContainerColumn::alignLeft |
                                 IContainerColumn::readOnly | IContainerColumn::horizontalSeparator);

// Create the Message Icon column
 ivpColMsgIcon=new IContainerColumn (offsetof(MsgVwrMessage, ivMsgIcon), IContainerColumn::string |
                                 IContainerColumn::alignVerticallyCentered |
                                 IContainerColumn::alignCentered | IContainerColumn::readOnlyHeading,
                                 IContainerColumn::icon | IContainerColumn::alignVerticallyCentered |
                                 IContainerColumn::alignLeft |
                                 IContainerColumn::horizontalSeparator);
```

```
// Create the Message Date column
ivpColMsgDate = new IContainerColumn (offsetof(MsgVwrMessage, ivMsgDate), IContainerColumn::string |
                                      IContainerColumn::alignVerticallyCentered |
                                      IContainerColumn::alignCentered | IContainerColumn::readOnlyHeading,
                                      IContainerColumn::date | IContainerColumn::alignVerticallyCentered |
                                      IContainerColumn::alignLeft |
                                      IContainerColumn::readOnly | IContainerColumn::horizontalSeparator);

// Create the Message Time column
ivpColMsgTime = new IContainerColumn (offsetof(MsgVwrMessage, ivMsgTime), IContainerColumn::string |
                                      IContainerColumn::alignVerticallyCentered |
                                      IContainerColumn::alignCentered | IContainerColumn::readOnlyHeading,
                                      IContainerColumn::time | IContainerColumn::alignVerticallyCentered |
                                      IContainerColumn::alignLeft |
                                      IContainerColumn::readOnly | IContainerColumn::horizontalSeparator);

// Set the title data for the Message Viewer container
ivpMsgVwrCnr->setTitle("Message Viewer");
ivpMsgVwrCnr->disableTitleUpdate();
ivpMsgVwrCnr->showTitleSeparator();
ivpMsgVwrCnr->showTitle();
ivpColMsgNumber->setHeadingText("Message Number");
ivpColMsgText->setHeadingText("Message Text");
ivpColMsgIcon->setHeadingText("Icon");
ivpColMsgDate->setHeadingText("Date");
ivpColMsgTime->setHeadingText("Time");

// Add column separators
ivpColMsgNumber->showSeparators();
ivpColMsgText->showSeparators();
ivpColMsgIcon->showSeparators();
ivpColMsgDate->showSeparators();
ivpColMsgTime->showSeparators();

// Add the columns to the Message Viewer container
ivpMsgVwrCnr->addColumn(ivpColMsgNumber);
ivpMsgVwrCnr->addColumn(ivpColMsgText);
ivpMsgVwrCnr->addColumn(ivpColMsgIcon);
ivpMsgVwrCnr->addColumn(ivpColMsgDate);
ivpMsgVwrCnr->addColumn(ivpColMsgTime);

// Set the split bar position
ivpMsgVwrCnr->setDetailsViewSplit(ivpColMsgText, 600);

// Remove all the objects in the Message Viewer container after closing
ivpMsgVwrCnr->setDeleteObjectsOnClose();

// Remove all the columns in the Message Viewer container after closing
ivpMsgVwrCnr->setDeleteColumnsOnClose();

// Display the Message Viewer Window
setClient(ivpMsgVwrCnr);
moveSizeTo(IRectangle(10, 250, 610, 450));
ivpMsgVwrCnr->showDetailsView();
show();
ivpMsgVwrCnr->setFocus();

} // End of MsgVwrWindow::MsgVwrWindow
```

```
//*****************************************************************************
// MsgVwrWindow - Destructor
//*****************************************************************************
MsgVwrWindow::~MsgVwrWindow()
{ // Start of MsgVwrWindow::~MsgVwrWindow

 // Delete the Message Viewer container
 delete ivpMsgVwrCnr;
} // End of MsgVwrWindow::~MsgVwrWindow


//*****************************************************************************
// MsgVwrWindow - addMessage
//*****************************************************************************
void MsgVwrWindow::addMessage(const MsgVwrMessage * pMessage)
{ // Start of MsgVwrWindow::addMessage

 // Serialize the message insertion
 IResourceLock MvwrLock(ivMutexSem);

 If(ivpMsgVwrCnr->objectCount() >= ivulMaxNumMsg) { // The maximum number of
            // messages has already been inserted in the Message Viewer window

  // Remove the first message in the Message Viewer window
  MsgVwrMessage * pFirstMsg= (MsgVwrMessage *)(ivpMsgVwrCnr->objectAt(0));
  if(pFirstMsg->isModal() &&
    (!(pFirstMsg->isProcessed()))) {
   pFirstMsg->setProcessed(0);
  }
  ivpMsgVwrCnr->removeObject(pFirstMsg);
 }

// Add the message as the last record in the Message Viewer container
 ivpMsgVwrCnr->addObject(pMessage);

 return;
} // End of MsgVwrWindow::addMessage
```

4. MSGVWR.RC LISTING

```
// ************************************************************************ //
//
// Module Name    :  MSGVWR.RC
//
//  Descriptive Name:  Message Viewer Context menus
//
// ************************************************************************ //

#include <os2.h>
#include <MSGVWR.RCH>

// Message Viewer main context menu

MENU ID_MENU_MSGVWR_MAIN
    BEGIN
      SUBMENU "~Help", IDM_HELP, MIS_TEXT
        BEGIN
          MENUITEM "Help ~index",IDM_HELPINDEX, MIS_TEXT
          MENUITEM "~General help", IDM_MSGVWR_HELP, MIS_TEXT
          MENUITEM "~Using help", IDM_HELPUSINGHELP, MIS_TEXT
          MENUITEM "~Keys help", IDM_HELPKEYS, MIS_TEXT
        END
      MENUITEM "~Select all", IDM_MSGVWR_SEL_ALL, MIS_TEXT
      MENUITEM "Deselect a~ll", IDM_MSGVWR_DESEL_ALL, MIS_TEXT
      MENUITEM SEPARATOR
      MENUITEM "D~elete", IDM_MSGVWR_DELETE, MIS_TEXT
    END


// Message Viewer object context menu
MENU ID_MENU_MSGVWR_OBJ
    BEGIN
      SUBMENU "~Help", IDM_HELP, MIS_TEXT
        BEGIN
          MENUITEM "Help ~Index", IDM_HELPINDEX, MIS_TEXT
          MENUITEM "~General help", IDM_MSGVWR_HELP, MIS_TEXT
          MENUITEM "~Using help", IDM_HELPUSINGHELP, MIS_TEXT
          MENUITEM "~Keys help", IDM_HELPKEYS, MIS_TEXT
        END
      MENUITEM "~Message help", IDM_MSGVWR_HELP_FOR_OBJ, MIS_TEXT
      MENUITEM SEPARATOR
      MENUITEM "~Delete", IDM_MSGVWR_DEL_ONE, MIS_TEXT
      MENUITEM "O~k", IDM_MSGVWR_OK, MIS_TEXT
      MENUITEM "C~ancel", IDM_MSGVWR_CANCEL, MIS_TEXT
    END
```

5. MSGVWR.RCH LISTING

```
#ifndef MSGVWR_RCH
#define MSGVWR_RCH

// ********************************************************************************** //
//
// Module Name   :  MSGVWR.RCH
//
//  Descriptive Name:  Include file for MSGVWR.RC. It contains all the
//                     definitions for the Message Viewer window context menus.
//
// ********************************************************************************** //


// Message Viewer Details view container ID
#define ID_CNR_MSGVIEWER          50

// General help items
#define IDM_HELP                  100
#define IDM_HELPINDEX                 110
#define IDM_HELPUSINGHELP             120
#define IDM_HELPKEYS                  130

// Message Viewer main context menu
#define ID_MENU_MSGVWR_MAIN       200
#define IDM_MSGVWR_HELP               210
#define IDM_MSGVWR_SEL_ALL            220
#define IDM_MSGVWR_DESEL_ALL          230
#define IDM_MSGVWR_DELETE             240

// Message Viewer object context menu
#define ID_MENU_MSGVWR_OBJ        300
#define IDM_MSGVWR_DEL_ONE            310
#define IDM_MSGVWR_HELP_FOR_OBJ       320
#define IDM_MSGVWR_OK                 330
#define IDM_MSGVWR_CANCEL             340

#endif
```

6. TESTMVWR.CPP LISTING

```
// ********************************************************************** //
//
// Module Name    :  TESTMVWR.CPP
//
//  Descriptive Name:  Test program for Message Viewer classes
//
// ********************************************************************** //

#include <ithread.hpp>                  // IThread
#include <iapp.hpp>                     // IApplication
#include <ihandle.hpp>                  // IPointerHandle
#include <istring.hpp>                  // IString

#include <msgvwr.hpp>                   // Message Viewer classes

// Local functions prototypes
void AddMsgInThread(void *);

// Global variables
MsgVwrWindow * pMessageVwrWindow;

// ********************************************************************
// Name : main
//
// Operation : Test the Message Viewer
//
// Input Parameters:  none
//
// ********************************************************************
void main()
{ // Start of main
 // Create a MsgVwrWindow global object and initialize pMessageVwrWindow
 pMessageVwrWindow = new MsgVwrWindow(500, 5);
 // Insert several messages in the Message Viewer window executing
 // the AddMsgInThread function for each message in a different thread
 IThread * pThread1 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                  (void *) (IBase::false),
                                  IBase::true);

 IThread * pThread2 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                  (void *) (IBase::true),
                                  IBase::true);

 IThread * pThread3 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                  (void *) (IBase::false),
                                  IBase::true);

 IThread * pThread4 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                  (void *) (IBase::true),
                                  IBase::true);

 IThread * pThread5 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                  (void *) (IBase::false),
                                  IBase::true);
```

```
IThread * pThread6 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::true),
                                 IBase::true);

IThread * pThread7 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::false),
                                 IBase::true);

IThread * pThread8 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::true),
                                 IBase::true);

IThread * pThread9 = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::true),
                                 IBase::true);

IThread * pThreada = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::false),
                                          IBase::true);
IThread * pThreadb = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::true),
                                 IBase::true);

IThread * pThreadc = new IThread((IThread::OptlinkFnPtr)AddMsgInThread,
                                 (void *) (IBase::true),
                                 IBase::true);

 // Get the current application and run
 IApplication::current().run();
} // End of main
```

```
// ****************************************************************************
// Name : AddMsgInThread
//
// Operation : Adds a message in the Message Viewer window. It is executed in
//             a separate thread.
//
// Input Parameters:  arg  - Flag to set the modality of the message
//
// ****************************************************************************
void AddMsgInThread(void * arg)
{ // Start of AddMsgInThread
 Boolean fModal=false;                          // Modality flag
 unsigned long ulResp=0; // Will contain the user response
 IString szMsg;        // Message string inserted
 MsgVwrMessage * pMsg = (MsgVwrMessage *)0; // Pointer to a MsgVwrMessage object
                                                // to be inserted in the Message Viewer window
 // Initialize fModal
 fModal = (Boolean) arg;

 If(fModal) { // The message is modal
  ISystemPointerHandle                                  // Icon associated with
    ModalMsgIcon(ISystemPointerHandle::question, true); // the modal messages
  // Set the message text
  szMsg = "Test modal error message from thread " +
          IThread::currentId().asString();

  // Create a MsgVwrMessage object
  pMsg = new MsgVwrMessage((IThread::currentId()),
                           szMsg,
                           ModalMsgIcon,
                           pMessageVwrWindow,
                           fModal);
 }
 else { // The message is non-modal
  ISystemPointerHandle                          // Icon associated with the
    NonModalMsgIcon(ISystemPointerHandle::warning, true); // non-modal messages

  // Set the message text
  szMsg = "Test non-modal error message from thread " +
          IThread::currentId().asString();

  // Create a MsgVwrMessage object
  pMsg = new MsgVwrMessage((IThread::currentId()),
                           szMsg,
                           NonModalMsgIcon,
                           pMessageVwrWindow,
                           fModal);
 }

 // Insert the created MsgVwrMessage object in the Message Viewer window
 ulResp = pMsg->addToMsgVwr();

} // End of AddMsgInThread
```

REFERENCES

OS/2 2.0 Programming Guide Volume I (IBM DOC. S10G-6261)
C/C++ Tools: User Interface Class Library User's Guide (IBM DOC. S82G-3743-00)
C/C++ Tools: User Interface Class Library Reference (IBM DOC. S82G-3738-00)
Making Concurrent Messages Under OS/2 2.x published in the OS/2 World Conference & Exhibition Proceedings (Volume 2), July 20-22, 1994 Santa Clara, California (Miller Freeman, Inc. - ISBN # 0-87930-347-6)
OBJECT-ORIENTED MODELING AND DESIGN - James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen - 1991 (Prentice Hall, Inc. - ISBN # 0-13-629841-9)

PRO 1008
Visual Builder for C Set++

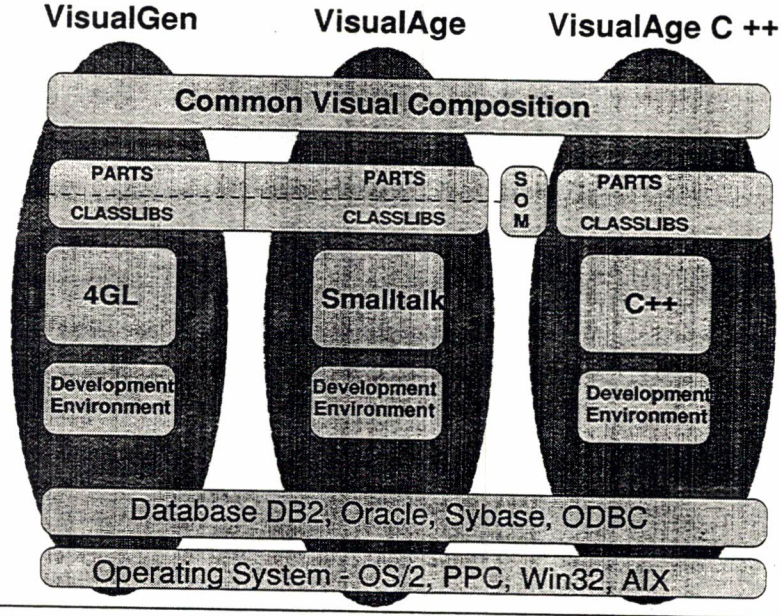
Dale Nilsson
IBM

Agenda

- VisualAge Family
- C Set Parts Architecture
- Visual and Non-visual parts
- Events, Actions, and Attributes
- Design considerations
- Demo
  - Visually building data access parts
  - Importing C++ classes
  - Layout of GUI
  - Making Connections
  - Generating C++ code
  - Toolbar
- Questions and answers

IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Visual Builder Solutions



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

VisualAge C++ V3 Components

Compiler

Compiler
- Performance Improvements
- Opt for Size, Debug Pack
- Pre-compiled headers

Direct to SOM
- Compiler support for DTS and IDL
- Debug and browse support

Linker
- 32-bit rewrite based on LINK386
- Smart Method Linking
- 2-4X performance Improvement

Runtime
- Posix Locale
- New memory mgmt

New Function for Version 3 shown in green italics

Tools

Visual Builder
- Full visual programming
- support for visual and non-visual parts

Data Access Builder
- Class libs and visual schema mapping tools for data access
- Generation of C++ source for non-visual parts

EXTRA!
- Interface rewrite

ICSDEBUG
- PICL based

LPEX
- Syntax highlighting editor built into environment

Browser
- Complete rewrite
- Performance improvements
- Simplified queries
- Source code browsing

Class Libs

Class Libraries
- UI: Toolbar, Notification Framework, Multimedia, 2D Graphics
- CC: part enablement

IDE

Workframe
- New Project windows
- Usability enhancements
- Tool to tool via router
- "C Set ++" pulldowns

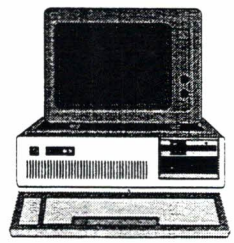
Documentation
- Task oriented - How do I?
- Universal access

Install
- Unified Install/2 based
- Toolkit incorporated

(C) Copyright IBM Corporation 1995 - All rights reserved

VisualAge C++ V3 Environment

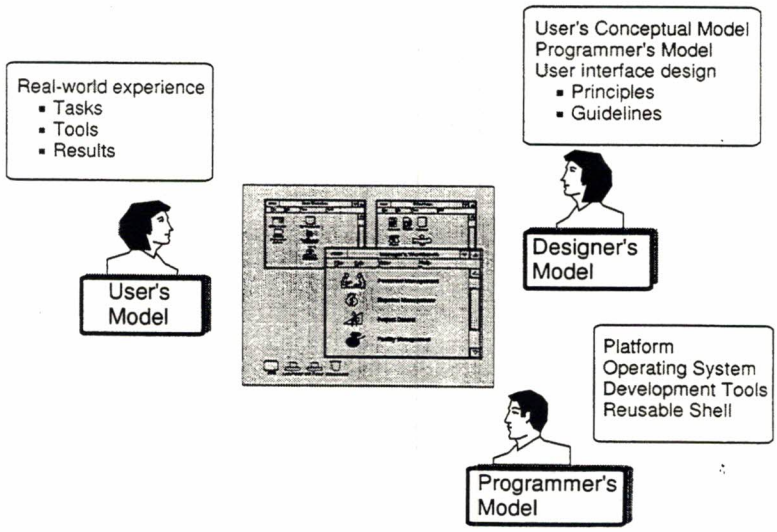
- **Packaged as a standalone single user version**
- **Supports OS/2 2.11 or OS/2 Warp**
- **Requires OS/2 Warp Toolkit**
- **Development environment**
  - **16mb minimum, 32mb recommended**
  - **33mhz 486 minimum, 66mhz 486 recommended**
  - **VGA minimum, SVGA recommended**
  - **DB/2 1.2 needed for data access**
  - **MMPM/2 needed for multi-media**
  - **SOM Developer's Toolkit for SOM**



(C) Copyright IBM Corporation 1995 - All rights reserved

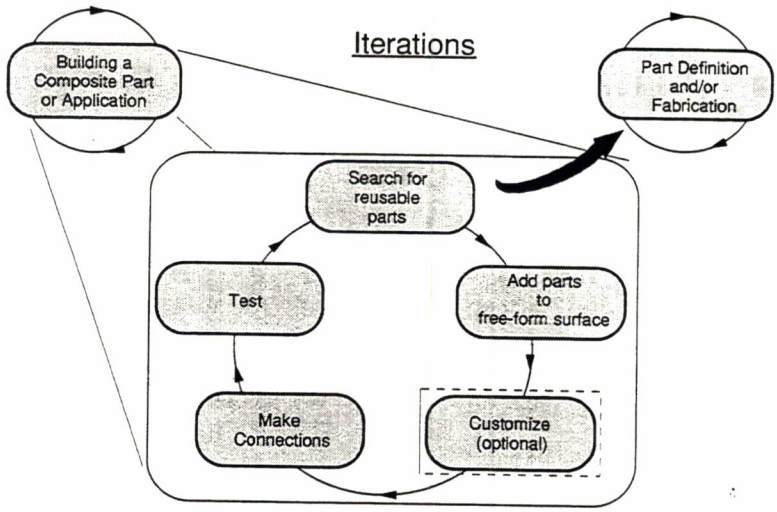
Designing Visual Classes

Factors influencing the User Interface design



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

The Process of Assembly



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

C ++ Parts Architecture

- Parts are C++ Classes
- Parts have an Event/Action/Attribute interface
- Parts have notification - subclass IStandardNotifier (IPart)
- Parts are either visual or non-visual
- Supports separation of model and view
- Makes encapsulation easier and promotes reuse

Events ***Attributes***

Actions

 (C) Copyright IBM Corporation 1995 - All rights reserved

Visual and Non-visual parts

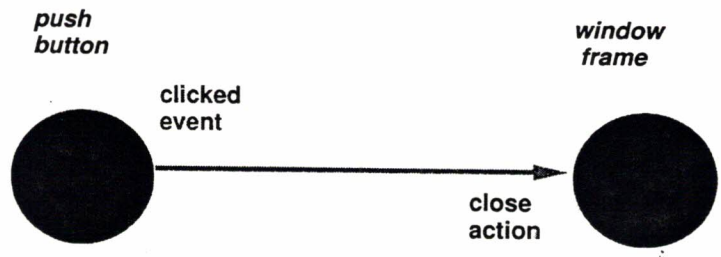
- Visual parts are:
  - UI components in IBM Open Class
  - Generated by VisualAge C++
  - Composite parts with at least one visual part

- Non-visual parts are:
  - Application logic in C++ like:
    - Data access
    - Business Logic
  - Maintained in separate code files
  - Represented as an icon in the composition editor

 (C) Copyright IBM Corporation 1995 - All rights reserved

Events

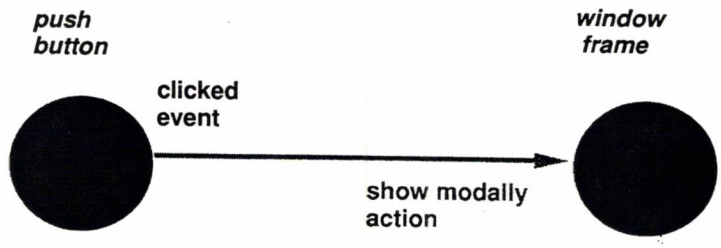
- Trigger notification to other parts
- Connect to Actions or Attributes
- Uni-directional connections



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Actions

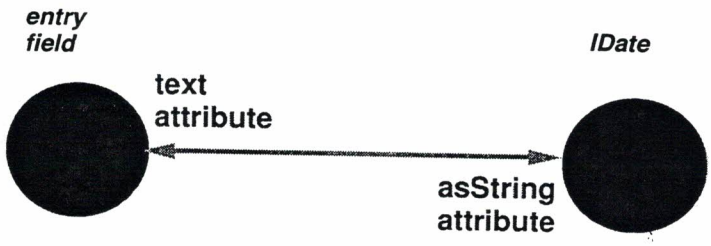
- Operations which a part can perform
- Public member functions
- Target of uni-directional connections



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Attributes

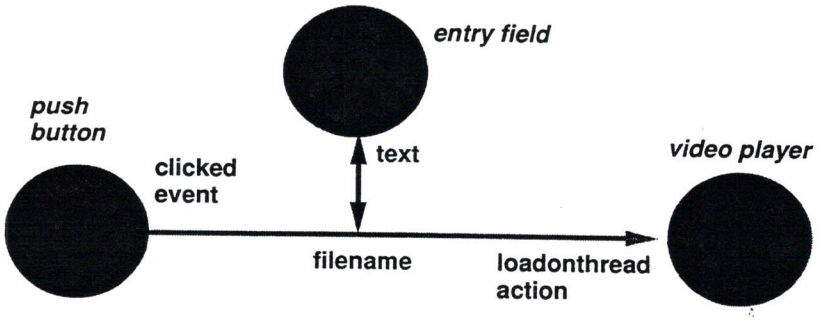
- Data that can be accessed by other parts
- Public data
- Bi-directional connection to other attributes



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Passing Parameters

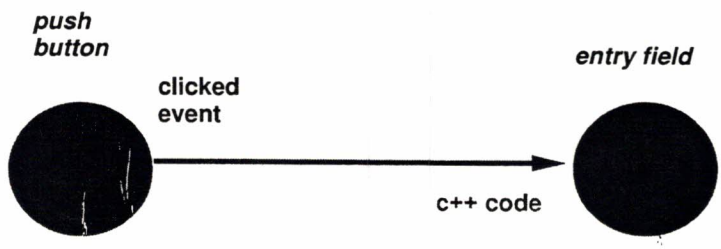
- Similar to Attribute to Attribute connections
- Connect to other connections
- Bi-directional connections



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Event to Custom Logic

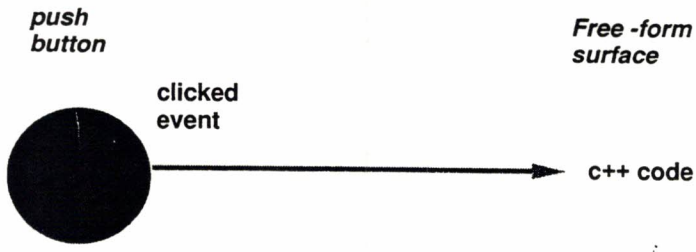
- Trigger notification to C++
- Attached to another part
- Uni-directional connections



IBM (C) Copyright IBM Corporation 1995 - All rights reserved

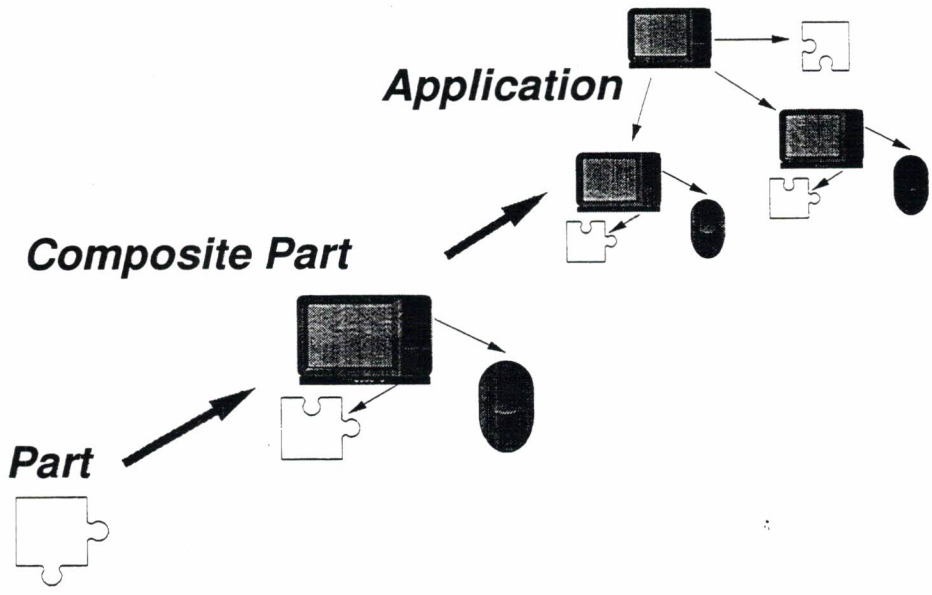
Event to Member function

- Trigger notification to C++
- Attached to the Free-form surface
- Uni-directional connections

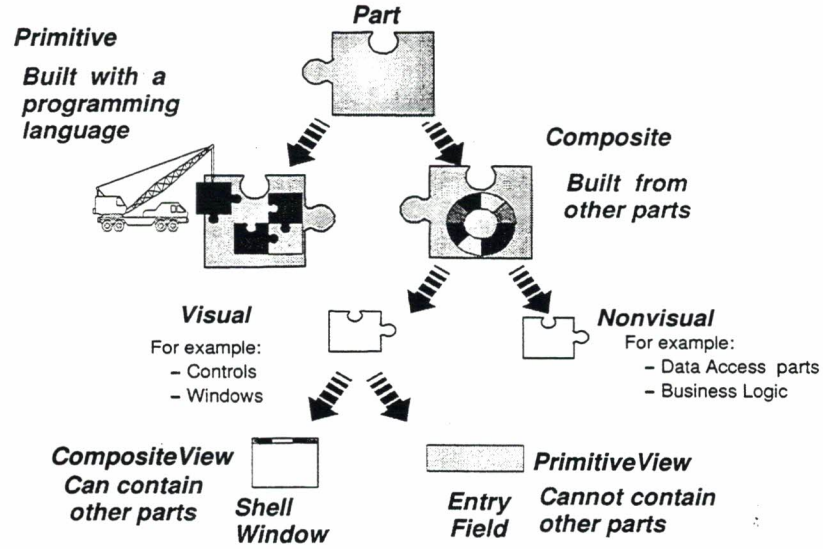


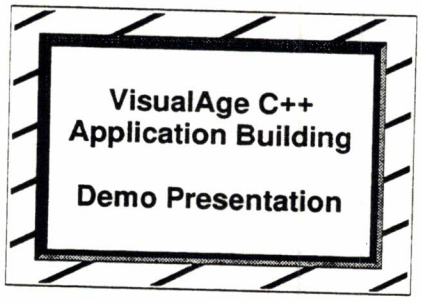
IBM (C) Copyright IBM Corporation 1995 - All rights reserved

Building an application



Types of Visual Builder Parts





Dale Nilsson

© IBM Copyright IBM Canada Ltd. 1993, 1994

Acronym Guide

AIX	Advanced Interactive Executive (IBM's UNIX)
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CUA91	Common User Access 91
DCE	Distributed Computing Environment
DOMF	Distributed Object Management Facility
DSOM	Distributed System Object Model
IPX / SPX	Novell Netware communication protocol
LAN	Local Area Network
OMG	Object Management Group
REXX	Restructured Extended Executor - procedural language included with OS / 2
RISC	Reduced Instruction Set Computer
SOM	System Object Model
SQL	Structured Query Language
TCP / IP	Transmission Control Protocol / Internet Protocol
UI	User Interface

F.W. Fang

© IBM Copyright IBM Canada Ltd. 1993, 1994

SYS 1002
Getting it Right: Testing OS/2 Applications

Phil Arthur
IBM

Philip L. Arthur
International Business Machines, Corp.
Dallas Systems Center
AD Consulting and Services
1 East Kirkwood Blvd.
Roanoke, TX 76299
Telephone: (817) 961-5494
Internet: arthur@vnet.ibm.com
IBM Mail: USIB2N2N at IBMMAIL
Copyright IBM Corporation, 1995

The author of this document welcomes reader feedback. The document content is based upon experience and consulting engagements within IBM and with IBM customers. Simplified examples are used to convey the concepts and stimulate ideas; however, the examples are based on more sophisticated implementations that are currently in use.

Testing OS/2 Applications

ABSTRACT

Studies show application testing consumes at least 50% of product labor cost. GUI applications have proven to be an even greater challenge for application developers and testers. This provides opportunities for automating and formalizing the test process and reducing cost while improving application development productivity, code quality and reliability. This article focuses on the process changes and testing theory for effectively implementing test automation. Automated test drivers test an application from an end user perspective by simulating end user keystrokes and mouse movements. Testing OS/2 applications is the primary focus of this article.

INTRODUCTION

At first glance, total life cycle costs of 50% for application testing seem much too high until you dissect the application testing process itself. Typical testing activities are creating test scenarios, identifying and creating input data, establishing test data bases, running the test cases, evaluating the test results, debugging/resolving code deficiencies, recompiling the program, and repeating one or more of the above activities until the code or application is deemed ready for production execution. Several consulting studies indicate that the average application program goes through some or all of these activities over 25 times before development is completed. That does not include production fixes, enhancements, and maintenance. Sophisticated applications involve test planning, design, validation, completeness and testing the entire application by separate test groups. These activities are often repeated for enhancements or code fixes with little reuse. Therefore test case reuse and maintainability are critical to improving testing productivity.

The application programmer controls application code quality in many application development organizations. The programmer receives requirements, writes the code, creates a few test cases, runs some unit test cases, probably uses a debug tool to run test cases and identify defects, and ultimately puts the code into production if the results "look good". Programmers tend to test the code they wrote; which may not match the application requirements. To test the code that should have been written requires developing test cases from the requirements. Detailed test plans are seldom defined prior to the coding phase. Therefore a rigorous testing process is necessary.

APPLICATION DEVELOPMENT PROCESS

Often the software development process has one quality checkpoint: after the product is built, the written application code is tested. A more effective process has quality checkpoints at each development phase. Software development lifecycle then becomes a set of work products and releases, permitting continuous assessment of product quality, and allowing for changes in the development process.

RIGOROUS REQUIREMENTS DEFINITION

Rigorous requirements definition allows one to focus on error prevention instead of error removal. B. W. Boehm states in Software Engineering Economics that the cost of error repair could increase by as much as 100:1, the later in the development process that an error is removed. Rigorous requirements definition provides a clear and concise set of programming specifications for the application programmer to write high quality code. Rigorous requirements definition, which includes data domain, data type, and allowed values information, allows application testers to build a rigorous set of test cases and validate the test results. Rigorous requirements allow one to build their test plan and software acceptance criteria during the analysis and design phase.

AUTOMATED TESTING

Given data domain, allowed values, and application process descriptions, one can create WITT's English-like test scripts prior to code completion. WITT is a computer-based CUA91 application test tool that tests applications from an end-user perspective.

SOFTWARE LIFE CYCLE

A condensed view of application development is requirements, design, implementation, and operation and maintenance. Application test is typically viewed as part of the implementation phase. To improve application quality and the testing process, testing activities should be performed as early in the life cycle as practical. During the requirements phase a test plan should be developed, which may include objectives, schedule, test types planned (eg. system test, acceptance test). The master test plan should be developed during the application design phase, and should include test coverage matrices, testing techniques, tool selection. Early in the implementation phase and before code completion, the detailed test plan should be developed and include test data and documented testcase procedures. Upon code completion, the test cases are executed and results analyzed. Ongoing testing expenses occur during the operations and maintenance phase, where the updating detailed test plans and regression testing should occur. A database refresh is often needed to automate regression testing results verification.

To improve testing productivity across the entire life cycle, especially during the operations and maintenance phase, reuse of testware is critical. Test cases should be reused and maintained for ongoing use by maintenance programmers, test databases routines should be reused to restore data to a known state (especially needed for record and playback automated test tools). This data refresh is valuable for results validation reuse. Each test case procedure should be definable and repeatable to isolate it from other test case procedures, and to facilitate testware reuse. See IEEE Standard for Software Verification and Validation Plans (1012-1986) for more details.

REQUIREMENTS DECOMPOSITION

Test coverage matrices created during the design phase are necessary to ensure testing of all application functions. These matrices are developed from the requirements specifications. The first step is to list the business functions to be tested. These business functions are decomposed into reports, screens, and transactions (business logic). The final step is to decompose each report, screen, and transaction into detailed tests; such as, data field specifications, data validation logic and computations, and exception handling.

For example, business functions could include adding, modifying, and querying customer information. Adding new customer function could be decomposed into sub-functions; such as, one or more screens, audit trail, and response time criteria. Each of these sub-functions needs to be decomposed into detail test plan specifications. For example, a screen for adding new customers could be decomposed into data entry validation and corresponding error messages; such as, validate name, address, and phone number. Boundary Value Analysis testing technique is valuable for creating detailed test plans. Boundary Value Analysis approach selects one or more data elements that identify the boundary of a partition or data domain. If the partition represents a range of values, one would test the minimum and maximum values. If the partition represents a set of values as in a table lookup, one would test the first and last table entries. Using the number of hours in a regular work week example, testing data values -1, 0, 40 and 41 form four desirable test cases, which may replace the need to test data value 30. It may be desirable to test the maximum and minimum values allowed by the transaction screen's input field, which may not be consistent with the database definition. Experience indicates that Boundary Value Analysis testing often has higher payoff than other techniques. Boundary Value Analysis applies equally well to output data by testing maximum output, exceeding maximum output, minimum output and no output.

GUI TESTING

GUI applications are generally perceived as being harder to test than traditional text based applications. Window sizes and positions change, fonts, font sizes and attributes may change from user to user, and bit map window compares may find a few pixels differing. From a "Black Box" testing (end use testing) perspective nothing has changed because data edits must still be performed, error messages sent, data specifications must be adhered to, and logic/computations must be verified. Detailed test plans still need to be developed.

Test tool vendors are proclaiming that GUI applications are harder to test, but in reality, test tool vendors find it harder to build test tools for GUI applications. To overcome the GUI challenge, GUI test tools must operate at the object level, not at pixel x,y coordinates. For example, pressing an application push button should not be mouse clicks at x,y coordinates on the screen, but be English-like and simple as in PUSH-BUTTON "OK", which is the method used by IBM's object oriented Workstation Interactive Test Tool. This object orientation requirement renders hardware-based test tools obsolete, including stress and performance testing. Software based test tools like WITT, require little additional memory and playback applications as fast or faster than a human operator.

Using an object oriented test tool makes testing GUI and host/text based applications similar. For example, PUSH-BUTTON "OPEN" is similar to PF keys, however, the syntax is longer for the GUI automated test case procedure. Many other similarities to host/text based applications exist. GUI menu bar options are analogous to a host application main menu, and GUI menu pull-down options are analogous to a second host application menu. GUI listbox equivalent is a host scrollable menu list. A data entry field is a data entry field, and a combination box is a data entry field with a scrollable menu. Black Box testing of GUI applications are not all that different from testing text applications; however, GUI is more attractive and OS/2 offers multi-tasking, multiple threads, etc.

TEST AUTOMATION

Test automation means different things to different people. Typically it is a term used by software vendors selling record and playback tools. The automation is that the tool simulates a human tester upon playback.

TEST AUTOMATION APPROACHES

There are basically three approaches to test automation: record and playback, computer assisted coding, and use of a test case generator. Record and playback method needs an operational system to record human tester keystrokes and mouse movements. If the application contains application generated keys; such as, order number, policy number, customer number, the recorded test case procedures will require editing the recorded key to playback the test case. This approach is often the most labor intensive when maintaining the recordings for future regression testing.

Computer assisted coding allows creating automated test case procedures with an editor or language sensitive editor. This assumes the test tool externalizes and supports editing test case procedures that are ASCII files, or the tool provides its own editor for creating test case procedures without recording. This approach is often the most labor intensive to create, and execute because it is similar to writing a program to test a program. A variation on this approach is "callable routines", which has common test case procedures that are called by driver test case procedures.

A test case generator is the ideal method for test automation. A test case generator creates executable test case procedures, which are run via a playback tool. A test case generator does not create test case procedures by recording simulated application testing. The big advantage is not having to maintain test case procedures. For example, a change in a screen or window merely requires defining the new screen or window to the generator, which generates a new set of test case procedures. Test case generators can be application specific or general purpose.

DATA SEPARATE FROM PROCEDURE MYTH

A test case generator overcomes the deficiencies of having data separate from the test case procedure. Some test tool vendors claim advantages for having data separate from test case procedures; however, the primary use was to overcome record and playback deficiencies, namely application defined keys. Having the data separate requires maintaining two files when the application changes. In addition, data fields separated from the test case procedure may appear similar and difficult to identify for editing.

SELECTION CRITERIA FOR RECORD AND PLAYBACK TOOLS

The ideal test automation tool should be fully programmable and an extension to a language (eg. 2/REXX, C). It should record and playback keystrokes and mouse movements. For host applications, the test tool should support host session emulators. The test case procedure language should be English-like and editable with any editor. The test tool should support the usual screen functions; such as, saving screens, comparing screens, saving and comparing strings, marking non-compare areas (eg. data and time fields), and printing saved screens for user documentation. In addition, time initiated batch procedures for unattended testing are desirable. This allows one to run a series of tests from a single command file. Lastly, the test tool should run in a multi-tasking environment, allowing the tester to do other work while the test case procedures are executing.

DEFINABLE AND REPEATABLE TEST ENVIRONMENT

A definable test environment has data bases populated with test data that are known by the programmer and for which the programmer has defined the expected results for the application transactions. A repeatable test environment has data bases repopulated with the same definable test data to establish a regression test environment. Establishing a definable and repeatable test environment is critical to maximize the value of an automated test environment. An approach to refreshing the application test data files/tables, prior to running test cases, is to copy baseline data (master data) to the files and tables, overlaying the existing data.

CRITICAL SUCCESS FACTORS

Based upon consulting engagements, there are several critical success factors for test automation. A definable and repeatable test environment is necessary to quickly validate expected results and often to rerun automated test case procedures. Developing and documenting detailed test plans during the analysis and design phases can be done in parallel and independent from development. Combining test activities by building a comprehensive function test coverage plan avoids duplication and reduces costs. Creating reusable testware provides critical regression test capabilities. The test tool and methodology should support ease of testware maintenance and reuse. The test tool should be easy to use, powerful and flexible. Lastly, setting management expectations that test automation is not as simple as many test tool vendors claim. Creating and maintaining testware requires an investment in test tools and tester labor.

CONCLUSIONS

Testing is a major portion of software development and maintenance, but testing related test tools and support structure are often under invested. Existing testing methodologies and techniques still apply, but some degree of test automation is recommended. GUI applications are not more difficult to test, just different; however, GUI test tools are more difficult for test tool vendors to develop because the tools need to be object oriented and fully programmable. IBM's Workstation Interactive Test Tool fulfills these requirements. Hardware based test tools are obsolete for stress and performance testing of GUI applications. Lastly, a definable and repeatable test environment is desirable for test automation.

REFERENCES:

1. Boris Beizer, Software Testing Techniques, Van Nostrand Reinhold, New York, NY, 1991.
2. B. W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1980.

SYS 1007
Migrating Mainframe Programmers to OS/2

John Montgomery
IBM
Endicott, New York

INTRODUCTION

Who remembers the good old days of mainframe programming? Life was simple. You knew where the data was and you knew where the programs were running. Screen design was a matter of filling in a 80x24 character matrix. The I/T department had all the expertise on the systems that ran and all the control over the programs running in them. Why would anyone want to leave such a secure environment?

That is the basic problem in migrating mainframe programmers to OS/2. Such a transition involves changes to the programmer's hardware and software tools, working procedures and thought processes. The benefits of application development in OS/2 - multitasking, powerful programming languages and tools, availability of a GUI, increased productivity, reduced cycle time, improved technical vitality for the organization - are not sufficient to motivate a mainframe programmer who either finds it difficult to make the required changes or simply refuses. After several years of attempting to encourage, assist and cajole several organizations to make this transformation, I have compiled a list of the Top Five Responses Mainframe Programmers Give When Asked to Migrate to OS/2:

5. Oh yeah, OS2 - it's just like UNIX except the slashes go in the other direction.
4. I'm going to play devils advocate for a minute ...
3. Have you talked to my manager about this?
2. Hey, how do I print that online documentation?
1. OS/2 is too complicated, I don't have the right hardware, I don't have time to learn anything new, I don't know how to get started, it doesn't fit in with what I do now, and I'm not willing to lose any function or productivity!

MAINFRAME SUCCESSES

It's useful to look back at some successful transitions that have been made in the mainframe world to see if any lessons can be drawn from them. Figure 1 shows some of the changes which have occurred in several different aspects of mainframe programming.

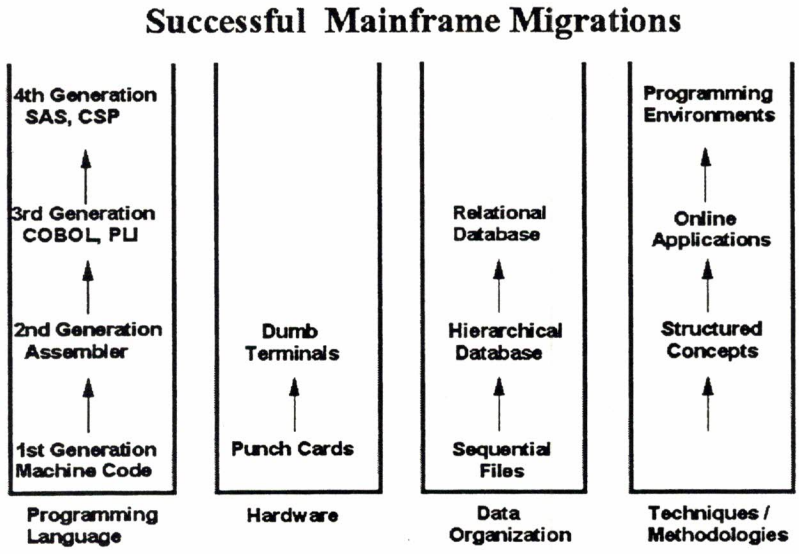


Figure 1

Languages used to build programs have progressed from machine code to high level, non-procedural products,

with several steps along the way. Each step required less lines of code and less dependency on knowledge of the underlying hardware. The hardware devices used in program development made a significant transition from punch cards to dumb terminals. Programming technology underwent huge changes as data organization evolved from sequential files to databases. New techniques and methodologies were constantly proposed and some have become industry standards such as structured programming and design concepts, online applications technology, and programming development environments such as ISPF.

Mainframe programmers who made these migrations had to deal with much the same fears of the unknown and technical problems as those facing the OS/2 migration. What attributes characterized these successful mainframe transitions?

- Long Transition Periods - None of these changes occurred overnight, but gradually evolved as successes with the new technology became more widespread. Each migration was spearheaded by "champions" within the programming organizations who were willing to take risks on something new and evangelize their successes to their peers. These champions were very often the youngest or newest members of their organizations; the ones who hadn't developed years of experience with the old ways. God created the world in six days, but he didn't have an installed base to migrate.
- Limited Scope - The mainframe migrations only affected one aspect of the programming environment. A programmer migrating from Assembler to COBOL, for example, only had the change the language being used, not the hardware and data storage as well.
- Easily Understood Benefits - Each of these migrations had a benefit which could be succinctly stated and measured as more experience was gained. The economic or productivity gains were visible, obvious and exciting to the participants and non-participants alike.

MOVING TO OS/2

These attributes are lacking in the mainframe to OS/2 migration due to the changing business environment and the very nature of the transition. These two factors combine to make the change both necessary and more difficult. Long transition periods are still required, but in an era of corporate downsizing (especially in I/T), less people are willing to take risks, the influx of new people into the organization has been curtailed, and it's harder to justify needed outlays of capital for new hardware, software and education. Migration to OS/2 is not limited in scope. All aspects of programming shown in Figure 1 are affected, most of them significantly. Many of the stated benefits of moving to OS/2 only become evident after a certain level of experience is reached and the uncertainties of the transitional state are overcome.

A key difficulty in moving established mainframe programmers into the OS/2 environment is that not only are the data and applications distributed, but the expertise is, too. No longer is the programmer the holder of all I/T knowledge. Easy-to-use products and the explosion of available PC literature have created "experts" all over the business organization. Productive business use of PC's did not originate in the I/T organizations, but in the financial groups. Their pioneering of spreadsheets is what transformed the PC from a hacker's tool to a business need.

John A. Zachman, formerly with IBM and now an independent consultant, describes his theory of change. He says that there are three options for effecting change in an organization to get them from State 1 to State 2:

- Increase the pain of State 1
- Decrease the uncertainty of the transition stage
- Increase the desirability of State 2

The last option, increasing the desirability of moving to an OS/2 programming environment, is the one which gets the most focus and which has thus far not overcome the technical, educational and emotional problems encountered. We need to spend more attention on increasing the pain of staying on the mainframe and decreasing the difficulty of getting to OS/2. Fear and reward are the basic reasons people do things; both need to be in our repertoire.

INCREASING THE PAIN

Increasing the Pain of State 1 reminds me of this story about my old Uncle Henry. Henry is not what you'd call a high-tech man; he doesn't believe in anything invented after 1960. He never saw the need for a VCR, an answering machine or a microwave oven. He won't use an ATM. In fact, if Uncle Henry can watch his two favorite TV shows, "Wheel of Fortune" and "Jeopardy" each day, he's totally content. One unfortunate day, the local TV stations juggled their schedules with the result that those two shows were broadcast at the same time on two different channels. What did Uncle Henry do? He bought a VCR immediately. Did VCR's suddenly get cheaper or easier to use? No, the pain involved in not having one got too great and the change was made.

In some ways, the job of increasing the pain involved in continuing to be a mainframe programmer is being done by external events. Nearly all the recent advances in application development technology are being offered exclusively on a PC and/or workstation platform. One only needs to check the skills being sought in want ads for I/T jobs to verify this.

In other areas, we can take advantage of the current business environment to increase the pain. The fear of losing one's job, as mentioned earlier, can work to discourage risk taking. However, survival can also be a powerful motivator. Sometimes, in order to achieve a change of the magnitude required here, it takes a devastating emotional experience such as watching the decimation of one's once proud mainframe department or the threat of such action. When faced with anarchy, many people will accept dictatorship.

Another way to inflict pain on those who refuses to even consider OS/2 is to offer something so wonderful on that platform that continued life on the mainframe seems bleak and unimaginable. A GUI or graphical editor will probably NOT be a big draw for these people. Try the Web Explorer. Find out what their main interest in life is and guide them to the corresponding place on the Web. I converted a life-long MVS bigot to OS/2 overnight merely by directing him to the New York Giants' Home Page. Once the OS/2 ice has been broken, transition to real work-related products is easier.

DECREASING THE UNCERTAINTY -

The task here is to counteract the resistance and fear which is rampant when people are asked to change the way they do their jobs. Change implies failure to some and those who prospered in the mainframe world are not anxious to see its demise. The persons in the organization responsible for making this change happen, the champions of the cause, have to be not only good technicians and planners, but also psychologists and baby-sitters at times. The strategy for overcoming this fear consists of Transition-Ware, a suite of OS/2 development tools appropriate for the organization, entry level projects on which to use the tools, and properly timed education.

DECREASING THE UNCERTAINTY - TRANSITION-WARE

The careful building of a suite of OS/2 development tools can go a long way towards decreasing the uncertainty of moving from the mainframe. Figure 2 shows two typical but very difficult migration paths.

A skilled and experienced mainframe programmer who has for years been using high level, non-procedural languages to create applications, does not spend much design or programming time trying to adapt to, or optimize

for, the existing hardware environment. These languages are called high level because they perform these hardware-dependent functions on behalf of the programmer. Migrating from such a 4th generation language on the mainframe to an OS/2 language like C (perhaps a 2.5 generation language in mainframe terms) requires skills long since forgotten of never learned. Pointers, memory allocation, and the primitive level of programming required when using Win... calls in Presentation Manager can be quite a cultural shock.

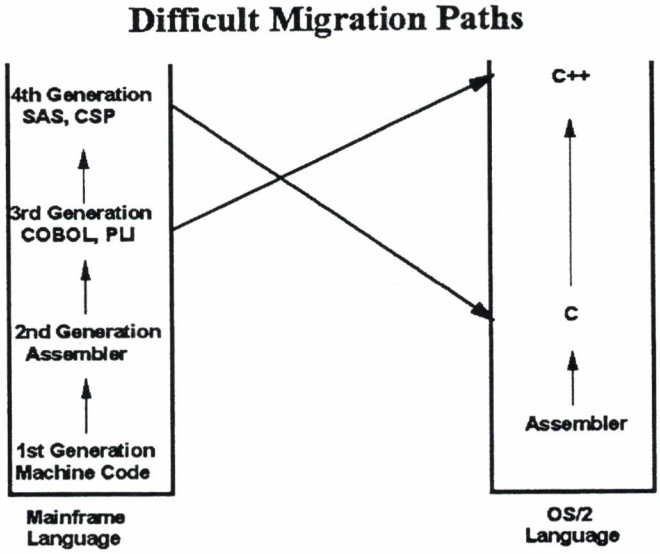


Figure 2

Moving up the programming evolution scale can be just as difficult. Migrating from mainframe COBOL to an object-oriented language like C++, requires enormous changes, not only in language syntax, but also in conceptual thinking about how to get the job done.

The paths depicted in Figure 3 provide an easier migration.

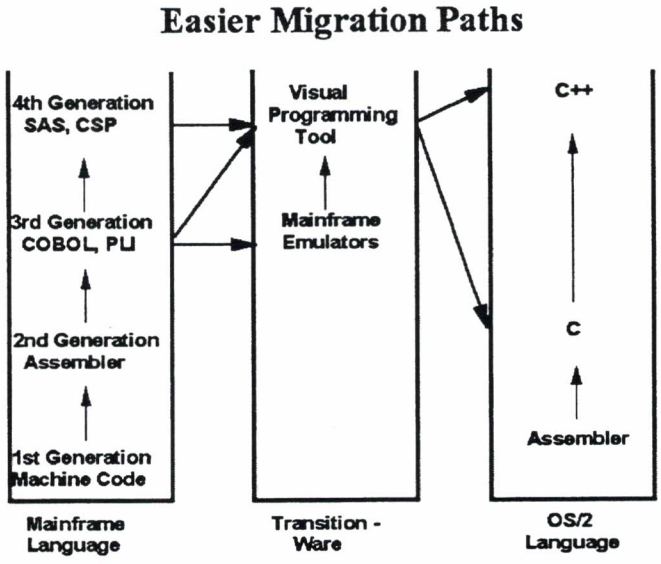


Figure 3

A group of tools which can be called "Transition-Ware" provide a stepping stone for mainframe programmers to begin development in OS/2 while taking advantage of their existing programming skills. Transition-Ware comes in two types:

- Mainframe Emulators - These tools allow migration of host skills with a minimum of training and emphasize what is similar on OS/2 rather than what is different. These tools can be as simple as the use of OS/2's EPM editor with a transparent upload/download capability or as complex as AD/MVS from MicroFocus. That product provides an exact OS/2 replica of MVS ISPF, down to the fake 'X SYSTEM' which appears in the lower left corner of the screen when the Enter key is pressed. COBOL, PLI, CSP and REXX programmers can be offered the same language on OS/2, DDCS and DB2/2 can provide transparent access to mainframe or OS/2 databases using familiar SQL calls, and mainframe-integrated library control can be achieved by the use of products like Endevor from Legent Corporation.

- Visual Programming Tools - Although these products do require some training and new skills, the transition time is relatively brief and mainframe programmers are able to create working (albeit simple) applications on OS/2 very quickly. The most effective type of visual programming tools for the OS/2 beginner are the ones which generate C or C++ code (e.g. Kaseworks). These products not only provide quick working applications, but are also a marvelous education opportunity. By using the visual portion of the tool to create changes in the application, the programmer can see the resulting change in the generated code, providing the basis for the understanding of the new language that will inevitably be needed when more sophisticated function is added to the application being built. Visual application builders such as VisualAge are a great introduction into object oriented and reuse concepts.

DECREASING THE UNCERTAINTY - SUITE OF OS/2 DEVELOPMENT TOOLS

A suite of OS/2 development tools offered to mainframe programmers should include Transition-Ware as well as a number of other "wares". It is important that the number of tools available to the organization be limited to those recommended by the group creating and supporting the suite. In the PC environment, it is much easier than on the mainframe for the organization to start into the Sinatra Methodology, where everyone does it "my way". One of the best OS/2 tools to assist in this area is the Shredder; get rid of all that junk that people have been collecting on their individual machines. Offer the suite as a set of icons in a folder which is easily installed on the OS/2 desktop. Use of a LAN-based offering eliminates the need for each programmer to worry about product installations, upgrades and DASD limitations.

The specific products offered are not as important as the types of products. They should accommodate the gamut of existing working environments on the mainframe. They should integrate easily with each other and with their counterparts on the host systems. Once the categories of needed tools are listed, a set of criteria should be agreed upon for the selection process. These criteria will vary according to the specific needs of the organization, but will probably include cost (both purchase and maintenance costs), existing expertise and experience, organizational strategic direction, required learning curve, runtime license requirements, compatibility with existing development environments, available support and vendor reliability. The suite suggested below is divided into two functional categories: those enabling migration of mainframe development and maintenance activity on OS/2, and those providing an environment for development of new OS/2 applications. The specific products listed are only examples.

Migration of mainframe activity on OS/2
  Utilities - Graphical editor (EPM), Hex editor, Language specific editors
  Online Help - (View, BookManager)
  Mainframe Emulators
    Languages - (MicroFocus COBOL, PLI for OS/2, OS/2 REXX)
    Library Control - (Endevor)
    Testing Tools - (SATT)
    ISPF Simulation - (AD/MVS)
    Host Database Access - (DDCS)
    Application Generators - (CSP/2)

Development of new OS/2 applications
- Languages - (C, C++)
- Workbench Environment - (WorkFrame/2)
- Testing Tools - (WITT)
- DBMS - (OS/2 DB2/2)
- OS/2 Toolkit
- Visual Programming Tools
  - Code Generators - (Kaseworks)
  - Application Generators - (VisualAge, VisualGen, Visualizer)

The suite should also contain an online directory which briefly describes each of the products and its recommended usage.

DECREASING THE UNCERTAINTY - PROJECTS AND EDUCATION

If you build it, will they come? Now that the suite of OS/2 development tools is created and made available, what happens next? Without any further encouragement, a select few self-motivated individuals will start using the tools. A certain percentage of this small group will be hackers who will play with some of the available products and then go on to other things, communicating their experiences to no one. An even smaller percentage will attempt to use the suite to do something constructive. These are the agents who need to be identified, nurtured, and encouraged to spread the word throughout the organization.

It's important that the first projects are small in scope, but "real" work which is important to someone. These could include an OS/2 front end to an existing mainframe system, or a spreadsheet with database access - something to initiate the front line into life on OS/2 and determine which products will benefit the larger organization. Follow-on projects might include putting together some common code segments or modules which are specific to systems being maintained on the mainframe, such as date conversion routines for your fiscal calendar. Each success should be widely publicized, with some visible management support. The suite of products and accompanying processes can then be refined through the results of experience.

While these entry level development efforts are underway, an education plan should be put in place. Use the knowledge of the existing mainframe programming environment to establish a migration path from each set of working processes, similar to those in Figure 3. The use of "desktop" education, modules which a programmer can go through at his/her own machine and pace, are very effective, as long as an expert is available to answer questions. Classroom education should be reserved only for kicking off large, staffed development projects. Consider bringing in outside help for such education. This can bring needed expertise and credibility to the education process.

Keep in mind that some of the skills needed in creating applications in the new world of OS/2, such as interpersonal and negotiating skills, are not easily taught. This may be thought of as more of an awareness process than a skills learning process.

ANOTHER LIST

Since we started with a Top Five List, let's finish with a Top Ten List. These are the Top Ten Steps To A Successful Migration To OS/2 From The Mainframe:

1. Understand the existing mainframe environment, processes and business drivers.
2. Identify the sources of resistance and fear.

3. Build an OS/2 development environment to fit the organization with appropriate Transition-Ware.
4. Uncover the agents of change within the organization.
5. Start small - be realistic about expectations.
6. Publicize success stories.
7. Gain management support.
8. Seed key projects with agents and the converted.
9. Supply education in the proper format and at the proper time. 10. Refine the OS/2 development tools and processes as experience dictates.

These Top Ten Steps may not be as entertaining as the Top Five list, but the results will be far more satisfying.

CS 1106
DB2/2 Database Administration

Rick Weaver
Mike Logan
IBM

Agenda:

- Database Systems Management - the challenge
- What needs to be managed
- Data Environments
- DB2 Family
- SystemView
- Installation
- Users and Security
- Performance and Tuning
- Event and Problem Monitoring
- SNMP Agent of DB2 for OS/2
- Data Replication Products

Agenda:

- DataHub for OS/2
- DataPropagator Relational
- IBM DataGuide
- IBM Visualizer
- IBM Visual Warehouse

Database Systems Management

- Need to manage:
  - Systems
  - Network
  - Database applications
  - Data
  - Integration of above
- The Challenge
  - Manage costs
  - Manage Complexity
- Solutions;
  - Built-in facilities
  - Add-on applications

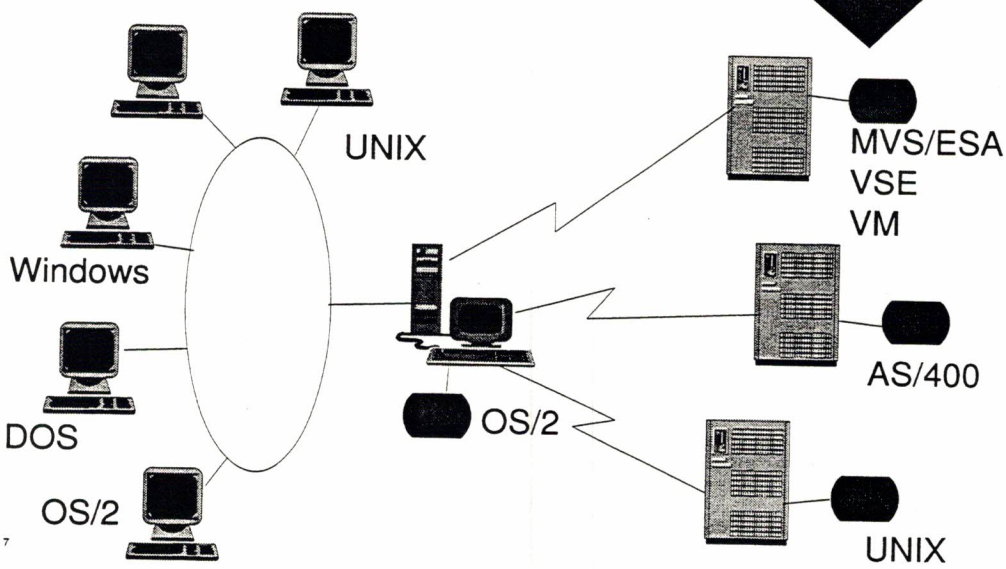
What Needs to be Managed

- Installation and update
- Configuration
- Users and security
- Events and problems
- Performance
- Backup and recovery
- Activity scheduling
- Data replication
- Data access

Environments

- Single System
- Departmental LAN
- Remote LANs
- Distributed WAN
- Host Connection

DB2 Family



SystemView

- IBM's strategic approach to systems management
- Emphasis on cross-platform support
  - Integration of applications
- Includes:
  - NetView
  - NetView Distribution Manager
  - ADSM
  - DataHub

Installation

- CID
  - Configuration, Installation, and Distribution
    - Centrally managed
    - Can store, distribute, and maintain software
- LAD/2
- NVDM/2

Users and Security

- Access to DB2/2
  - Authentication process
  - Managed outside DB2/2 by User Profile Management
  - Keep backup of NET.ACC file
- Access within DB2/2
  - Managed by DB2/2
  - Based on authorizations and privileges
- Authority levels determine privilege level
  - User
  - Local Administrator (DBADM)
  - Administrator (SYSADM)
- IDs can be maintained individually or as Group IDs

Performance and Tuning

- Utilities
  - REORGCHK
  - REORG
  - RUNSTATS
- EXPLAIN
  - Visual Explain - new with V2.1
    - See access plan
    - Tune SQL statements
    - Determine source of problems in transactions

Event and Problem Monitoring

- Database Systems Monitor - new to V 2.1
  - Comprehensive monitoring tool
  - Information collected:
    - General information
    - Database connections
    - Locks and deadlocks
    - Buffer pool activity
    - SQL sttatements
    - Cursors
    - Communication activity
    - Table and database activity
    - Unit of work status

Database System Monitor

- Snapshot Monitor
  - Collects point-in-time data
- Event Monitor
  - Query status of a database over a period of time
  - Summary information for transactions
  - Record trace of SQL statements for a transaction
- Performance Monitor
  - Show performance data graphically

Administrator's Toolkit - V2.1

- Integrated group of DBA tools
  - Performance Monitor
  - Visual Explain
  - Database Director
- GUI interface scales easily to DataHub

Diagnosing Problems

- OS/2 FFST/2 error logging
  - Error detection at the point of failure
  - Immediate problem notification
  - Unique problem identification
  - Faster problem diagnosis and resolution
- SQLTRACE
- On-line messagess

SNMP Agent

- New to DB2 for OS/2 V2.1
- Integration of network and database management
- Single image of database and systems management
- Standards-based management of DB2
- Supports the following standards:
  - RFC 1157: A Simple Network Management Protocol
  - RFC 1687: A Management Information Base for management of relational database management systems
- Requires one of the following:
  - DPI 1.1 support provided by TCP/IP V2.0
  - DPI 2.0 support provided by WARP Connect

Backup and Recovery

- Invoke from Recovery Tool, CLP, programs
- Backup entire database, or, table space (V2.1)
- SYSADM or DBADM authority required
- Automated backup and recovery
- Destination:
  - Local disk or LAN drive,
  - Destination specified in User Exit
  - ADSM (V2.1)
- V2.1 - Can initiate backup procedure from a client
- V2.1 - Backup online as well as offline
- V2.1 - Parallel backup and recovery
- Recovery history file

Rollforward Recovery

- Uses backup, applies logfiles to recover to
  - point in time, or
  - end of logfiles
- Uncommitted transactions are rolled back
- Minimal loss of transactions after an error
- To enable:
  - Enable database configuration parameter 'logretain'
  - Backup database to provide starting point for recovery
- To perform:
  - Restore database from backup
  - Retrieve logfiles and roll forward

DataHub

- Distributed database management
- Control point on either OS/2 or UNIX-based workstation
- Powerful graphical interface
- Will manage the following RDBMS's:
  - DB2 for OS/2
  - DB2/400
  - DB2 for MVS
  - DB2 for VM
  - DB2/6000

DataHub...

- DataHub functions will manage the following:
  - Objects
    - Browse catalogs
    - Create, drop, alter tables and views
  - Operations
    - Issue database and system commands
    - Run utilities
    - Schedule routine tasks
    - Monitor operating system resources
  - Authorizations
  - Storage
- Provides a launch platform for integrating tools

DataHub/2 - Flowmark Enabled

- Use Flowmark to define your workflow
- Drag-and-drop DataHub commands to workflow
- Interactively define workflow
- Tesst workflow with graphical animator
- Conditionally execute workflow steps
- Analyze result of workflow execution

DataPropagator Relational

- Launched from DataHub
- Provides refresh and update replication
  - From DB2 sources on MVS, OS/400, AIX, OS/2
  - To any DB2 target
- Can specify intervals for update
- Automated data copy
- Data Staging
- Open architecture
  - DRDA
  - SQL
  - Relational control structures

Other DataHub Enabled Tools

- IBM tools
  - Data Refresher
    - Refresh data from legacy sources to DB2
    - Data Propagator Nonrelational
  - DB2 Estimator
  - DB2 Performance Monitor
  - DB2 Visual Explain
- Vendor Tools
  - Bridge Fastload
  - Infotel INFORECOVERY
  - Legent IN2ITIVE
  - PLATINUM Quick Copy
  - Others

Data Guide for OS/2

- An information catalog for end-users
- Data Guide User
  - Provides search and browse capability
  - Can launch applications to view information quickly
- Data Guide Administrator
  - Provides extractors to gather information
    - Multiple sources

Visualizer

- Family of end-user products
  - Supports business information analysis
  - Decision support
- Query product
  - Database visualization
  - Ad hoc reports
  - Access to data
- Expandable toolset
  - Charts
  - Statistics and plans
  - UltiMedia Query
  - Development

MNC 1108
Creating a CID-Enabled Installation Using Software Installer

Ken Greenlee
IBM

Agenda

Introduction
CID Overview
Why Software Installer?
Software Installer Overview
Key End-user Functions
Attended Install Demo
Enablement Questions
Enablement Steps
Unattended Install Demo
CID-enablement - Developer Tasks
Solution Summary
Software Installer Functions (Notes)

Introduction

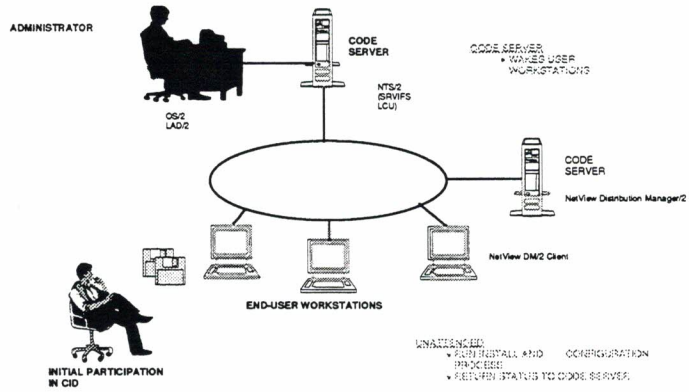
The number of workstations installed in organizations has grown steadily over the past 10 years. In addition, operating systems and application software have become larger and more complex. The process of installing and configuring software and data files is a tedious task that causes many users to delay the installation of new systems and the upgrading of existing systems.

Customers require a simple method to configure and install software and distribute it to their end users. In working with customers and vendors on this problem, IBM has developed a configuration, installation, and distribution strategy called CID, as an industry solution for customers to install, configure, and distribute their software.

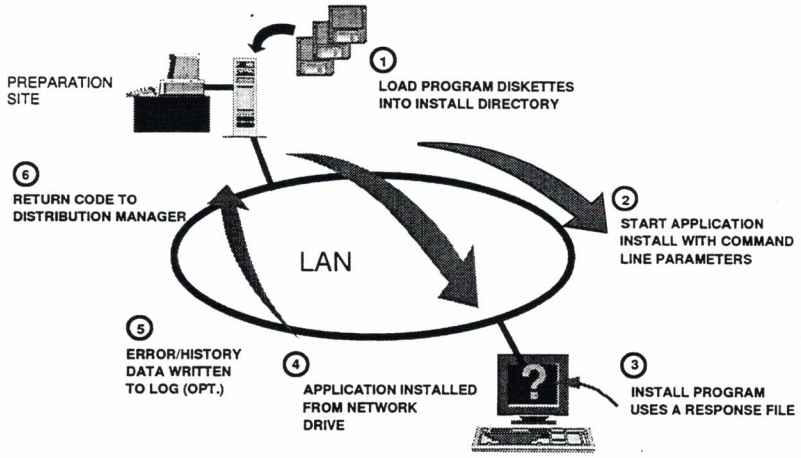
The CID architecture is part of the IBM distribution systems management strategy. Software Installer in conjunction with Netview DM/2 are two of the strategic software management tools chosen for implementation of this strategy.

CID Overview

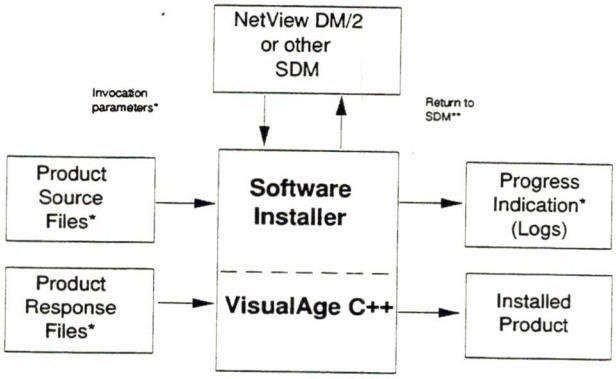
- One Time
  - Attended install on administration system
  - Copy Software images into code server library
- Workstation Tailoring
  - Tailor and validate response files
  - Select software to be distributed
    - Who gets what
  - Build and invoke install instructions
- Start automated process



CID-enablement Requirements



Software Installer CID Positioning



* Required to be CID enabled

Why Software Installer?

Software Installer is actually imbedded within your application. Software Installer implements the functions required to be a CID enabled application. For example, we support the command line parameters specified in the architecture. Software Installer imbedded in your application is designed to work in conjunction with a SDM. For example, IBM's Netview DM/2 can perform unattended distribution of CID enabled applications.

Software Installer Overview

Software Installer supports installation from:
- CD-ROM
- LAN
- Diskette

Software Installer supports installation from the host:
- Multiple Virtual Storage (MVS)
- Operating System/400 with IBM PC Support/400
- Virtual Machine
- Virtual Storage Extended (VSE/ESA) Version 1.3.0

Software Installer is currently translated into 12 languages.

Software Installer Key End-user Functions

Install your product in an unattended mode
Maintain your product by:
- Updating an installed product
  - *apply corrective service*
  - *apply a new release*

Restoring a previous release of a product
Deleting a product from the workstation
Install one product while using another
Selectively install components of a multi-component product

Attended Install Demo Notes

Enablement Questions

What media will you do distribution from (diskette, CD-ROM, LAN or Host)?
Answer: LAN

Will your product be packaged as one or more products?
- Answer: One

How many separately installable components make up your product?
- Answer: Ten

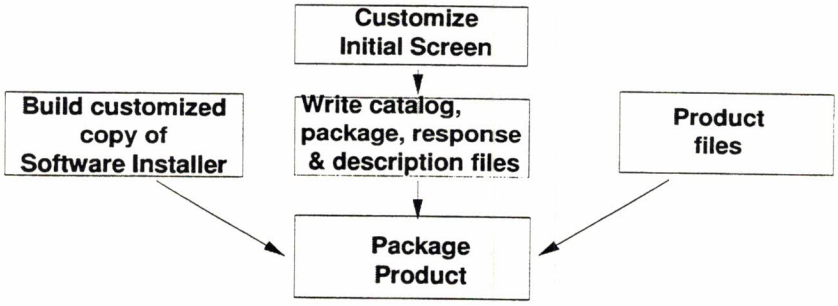
How many directories will my product be installed to?
- Answer: One

Will my product require changes to the CONFIG.SYS or OS2.INI?
- Answer: Yes

Does my product have any WorkPlace Shell objects to register?
- Answer: Yes

What do I want the end-user to see when the installation starts?
- Answer: Animated Bitmap of moving diskette

Software Installer Enablement Steps



Software Installer Enablement Steps

Create Catalog File
Create Package File
Create Description File
Build the hook .DLL (optional)
Modify, compile, and test the resource file
Create an INSTALL.IN_ file
Create the response file
Package the Product
- Diskette
- CD-ROM
- LAN

Test CID Enablement

Software Installer Package File Example

```
SERVICELEVEL
  LEVEL  = '010300_19950403'

DISK
  NAME = 'Software Installer - Diskette 1',
  VOLUME = 'SIOS2001'

PATH
  FILE = 'C:/IBB',
  FILELABEL = 'Installation directory'


COMPONENT
  NAME = 'Install first',
  ID = 'INSFIRST',
  DISPLAY = 'NO',
  SIZE = '1260000',
  SHARED = 'NO',
  DELETE = 'YES'
```

Software Installer Package File (Cont'd)

```
* READ.ME shadow in main folder

FILE
  EXITWHEN = 'INSTALL || UPDATE || RESTORE',
  EXITIGNOREERR = 'NO',
  EXIT = 'CREATEWPSOBJECT WPShadow "READ.ME"
<EPFIFLD> R
"SHADOWID=%EPFIFILEDIR%//READ.ME;OBJECTID=<EPFREADME>;"'

FILE
  EXITWHEN = 'DELETE',
  EXITIGNOREERR = 'YES',
  EXIT = 'DELETEWPSOBJECT <EPFREADME>'
```

Software Installer Package File (Cont'd)

```
FILE
  WHEN = 'OUTOFDATE',
  REPLACEINUSE = 'I U R D',
  SOURCE = 'DRIVE:EPFISCAT.ICF',
  VOLUME = 'SIOS2001',
  PWS = 'EPFISCAT.ICF',
  DATE = '950330',
  TIME = '1543',
  SIZE = '1020'

* Main product folder
FILE
  EXITWHEN = 'INSTALL || UPDATE || RESTORE',
  EXITIGNOREERR = 'NO',
  EXIT = 'CREATEWPSOBJECT WPFolder "Software Installer^ v1.3 for OS/2"
     <WP_DESKTOP> R          "ALWAYSSORT=YES;
ICONFILE=%EPFIFILEDIR%//EPFICLSD.ICO;ICONNFILE=1,
%EPFIFILEDIR%//EPFIOPEN.ICO;OBJECTID=<EPFIFLD>;"'
```

Software Installer Package File (Cont'd)

```
COMPONENT
  NAME = 'Install Utility: English (ENU)',
  ID = 'ENUNLV',
  REQUIRES = 'INSFIRST INSDOC DELLAST',
  DISPLAY = 'YES',
  DESCRIPTION = 'Select  "Install Utility: English (ENU)"  component.',
  SIZE = '690000',
  SHARED = 'NO',
  DELETE = 'YES'


FILE
  WHEN = 'OUTOFDATE',
  REPLACEINUSE = 'I U R D',
  UNPACK = 'YES',
  SOURCE = 'DRIVE:ENU\INSTALL.EX_',
  VOLUME = 'SIOS2002',
  PWS = 'ENU/INSTALL.EXE',
  DATE = '950331',
  TIME = '1459',
  SIZE = '129914'
```

Resource File (IIRC.RC)

A BITMAP line for each bitmap file you want to display in the initial window of your installation dialog should be defined to Software Installer. You can define a maximum of 100 bitmaps.

```
#include "iirch.h"

BITMAP  INST_BITMAP1    iircinst.bmp
BITMAP  INST_BITMAP2    iircarw.bmp
BITMAP  INST_BITMAP3    iircexcl.bmp
```

Resource File (IIRC.RC)

The following string table is used to customize the bitmap positions and movement characteristics in the initial installation window.

Coordinate parameters are:

initx	initial horizontal position of the bitmap
y	vertical position of the bitmap
endx	horizontal position you want the bitmap moved to
mvinc	number of picture elements to move the bitmap at one time
slices	number of horizontal segments to divide the bitmap into
delay	amount of time in tenths of a second to wait before drawing the next bitmap

```
STRINGTABLE
  BEGIN
    INST_BITMAP1_COORDS, " 1000,  15,   20,   7,    0,   5 "
    INST_BITMAP2_COORDS, "   -1,  62,   20,  20,    0,   5 "
    INST_BITMAP3_COORDS, "  330,  15,  330,   0,  100,  30 "
  END
```

Resource File (IIRC.RC)

Information window 1 is displayed after initial bitmap.

```
STRINGTABLE
  BEGIN
     INFO1_DISPLAY, "1"       /* display = 1, do not display = 0 */
     INFO1_TITLE,   "Instructions"
     INFO1_COORDS,  "13, 130, 520, 280"   /* x, y, width, height */
     INFO1_FILE,    "INFO1.TXT"
     INFO1_TEXT1,   "Welcome to <Product Name>. "
     INFO1_TEXT2,   "\015\015The following windows will guide you through the
      installation of <Product Name>."
     INFO1_TEXT3,   "\015\015Select Continue to start the installation.
      Select Exit to stop the installation."
  END
```

Resource File (IIRC.RC)

Depending on the value of the INST_SOURCES string, several windows are displayed that prompt for installation information.

```
STRINGTABLE
  BEGIN
    INST_DESTINATION,   "C:\\INSTDIR"
    SOURCE_DIALOG_POS,  "0, 100"
    PROGRESS_DIAGLOG_POS, "0, 100"
  END
```

The system sources are specified in the following string table.

```
STRINGTABLE
 BEGIN
    INST_SOURCES,  "DRIVE"
 END
```

Resource File (IIRC.RC)

Define the appearance and characteristics of the main window :

```
STRINGTABLE
  BEGIN
/*                                      x      y      width    height  */
/*                                     ---    ---    -------  -------  */
    EXIT_BUTTON_COORDS,       "542,   325,    75,      35"
    INST_WINDOW_SIZE,         "630,   470"
    INST_WINDOW_TITLETEXT,    "Installation"
    INST_START_PD_TEXT,       "~Start install..."
    INST_BACKGROUND_COLOR,    "darkcyan"
  END
```

Resource File (IIRC.RC)

The startup parameters shown below in the string table are used by the installation program executables when your install is started. They provide the catalog file name and location, the product name, the action to be performed, the initial position of the Installation and Maintenance window, and the workstation name of the response file.

```
STRINGTABLE
  BEGIN
    STARTPARM_CATALOG_DRIVE, "CATALOG.ICF"
    STARTPARM_PRODUCT,      "<Product Name>"
    STARTPARM_ACTION,       "i"
    STARTPARM_POSITION,     "10, 10"
    STARTPARM_RESPFILE,     "RESPONSE.RSP"
  END
```

Unattended Install Demo Notes

CID-enablement (Developer Tasks)

Package File
- User Exits
  - *Reboot is required*
- Return Codes

Response File

Documentation

CID-enablement tests
- command line
- Software Distribution Managers (SDMs)
  - *Netview DM/2*
  - *LCU*

What's in a Response File?

```
*-------------------------------------------------------------------
* Define the target directories.
*-------------------------------------------------------------------
FILE=C:\PRODNAME

*-------------------------------------------------------------------
* Include a COMP keyword for every component in the package file.
*-------------------------------------------------------------------
COMP=Component 1
COMP=Component 2

*-------------------------------------------------------------------
* The following 4 keywords are required.  See the Software Installer
* Reference for possible values.
*-------------------------------------------------------------------
CFGUPDATE=AUTO
OVERWRITE=NO
SAVEBACKUP=NO
DELETEBACKUP=NO
```

What's in a Response File? (Cont'd)

What if I have a keyword that my product uses, but is not supported by Software Installer. Can I still put it in my product's response file?

The answer is YES. You must put any information that a user would supply to your product through a dialog and any installation variables needed by your product's installation processing. Software installer will set an installation variable with your keywords in it. For example, MONITOR=VGA creates an installation variable with the name MONITOR and a value of VGA.

Solution Summary

There are significant advantages to CID-enabling products rather than using diskette installation:

- Faster rollout
- Unattended operating at the target machine
- Less system skill is required from the user
- Hardware and software changes can be accomplished without requiring users' skills to be upgraded
- By design, CID-enabling works for networks of both heterogeneous and homogenous systems.

Where to Obtain More Information

Product information can be obtained through the following channels:

- Technical Sales Support Fax: 919-254-4820
- IBM SWS Development Lab
  - *US: 1-800-426-2279*
  - *Outside US: 1-919-254-4760*

CompuServe: GO OS2DF2, Section 8.
For CompuServe registration and access questions, call:
- US: 1-800-848-8990 (representative #239)
- UK: 0800-289-378
- Germany: 0130-86-4643

Software Installer is available through The Developer Connection for OS/2 by calling:
- US: 1-800-6-DEVCON

Software Installer Functions (Notes)

Create a standard installation procedure
- Attended
- Lightly-attended
- Unattended

Create CID-enabled installations
Build distribution diskettes
Compress product files
Decompress packed files during install
Run application-specific exits

Software Installer Functions (Notes)

Tailor the initial installation screen to:
- Display up to 100 bitmaps
- Animate bitmaps on the initial installation screen
- Display an option info window to the user during install

Replace product files that are in use
Replace system files in use (available in release 1.3)
Change the text for help panels

Software Installer Functions (Notes)

Run exits supplied by Software Installer that can:
- Set environment variables
- Modify the user's CONFIG.SYS files
- Add and delete information from OS2.INI
- Add and delete information from the application's INI file
- Register and deregister Workplace Shell object classes
- Create and delete Workplace Shell objects

Run application-specific exits
Prompt the user for up to 20 install directories

Additional Function Notes

CD-ROM

Enabling your product for CD-ROM is just like enabling it to install from a single diskette. You will need two package files if you plan to ship on both CD-ROM and diskette.

Maintenance Functions

Software Installer maintenance functions can also be provided so that your customer can update, restore and delete your application.

User Exits

You can write user exits to further customize your installation for application specific tasks. Software Installer will invoke your user exits during installation. You can use any language, including REXX, to write the exits.

Inter-exit Communication

If you have multiple user exits which you'd like to share information, Software Installer provides APIs to allow you to store and retrieve data collected in the exits

Additional Function Notes (Cont'd)

CONFIG.SYS Manipulation

Software Installer provides you a lot of flexibility in modifying the CONFIG.SYS for your product. The ADDCONFIG and UPDATECONFIG entry types provide powerful adding and replacing capabilities. The Software Installer delete will even undo the CONFIG.SYS updates.

User-supplied Pack Routines

Software Installer provides an integrated pack routine to compress and decompress your files for installation. You also have the option to integrate your own pack routine. For example, several customer's have chosen to use PKZIP.

PRO 1105
Understanding & Exploiting OS/2 Graphical Programming

Kelvin Lawrence
IBM

Notices

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries.

- *IBM*
- *OS/2*
- *Operating System/2*
- *Presentation Manager*
- *WIN-OS/2*
- *Workplace Shell*

The following terms are trademarks of their respective companies.

- *Windows* *Microsoft*

Copyright © IBM Corporation 1994, All rights reserved.

Today's Agenda

- Quick GPI overview
- A word on presentation spaces
- Key concepts with working examples:
  - Geometric widelines
  - Polygons, arcs and full arcs
  - Coordinate spaces and transforms
  - Working with fonts
  - Paths and clipping
  - Palettes and bitmaps
  - A few words on performance

Reference Material

- OS/2 Presentation Manager GPI. By Graham Winn. Published by Van Nostrand Reinhold. . New (second) Edition now available.
- Presentation Manager Programming Reference, Volume 1. (Part of OS/2 Tech. Library) S10G-6624. Now superceeded by the OS/2 Warp versions.

Reference Material

- Programming the OS/2 Warp GPI by Stephan Knight. and Jeffrey Ryan Published by John Wiley.
- OS2DF1 (OS/2 Developers) forum on CompuServe
- Articles in PC Magazine by Charles Petzold and others
- Developer Connection News and CD-ROM

Reference Material

- OS/2 Developer magazine
- IBM International Technical Support Center "Red Books"
- Or.....You can reach me (Kelvin) via:
  - CompuServe: 72360,3554
  - Internet: lawrence@vnet.ibm.com

Sample programs

Please have a floppy ready if interested.

Samples available include:

Widelines (widein.exe)
Arcs/Pies (pie.exe)
Transformations (xform1.exe & xform2.exe)
Splines (spline.exe)
Fonts (font1.exe)
Multiple threading graphics (rotline.exe)
Paths/Fills/Clipping (clipfnt.exe)
Palettes (pmpal.exe)

What Is the OS/2 PM GPI ?

- GPI = Graphical Programming Interface
- Rich set of graphics routines for applications to utilize
- Largely device independent
- Fully 32-bit

The GPI also provides.....

- A very comprehensive 2-D imaging model.
- A lot more than we can talk about in one and a half hours!

GPI Function Classes

You can think of the GPI as providing various classes (or categories) of graphics functions.

By combining one or more categories of these functions we can quickly and easily build sophisticated graphics into our OS/2 Presentation Manager applications.

GPI Function Classes

- The GPI provides functions for:
  - Setting/querying attributes
    - *Colors*
    - *Linestyles*
    - *Fills, etc.*
  - Drawing basic primitives
    - *Lines, arcs and curves*
    - *Boxes*
    - *Polygons*
    - *Bitmaps, etc.*

GPI Function Classes

- The GPI provides functions for:
  - Building more complex shapes
    - *Regions*
    - *Areas*
    - *Paths*
    - *Clipping*
  - Transforming data
    - *Translate*
    - *Rotate*
    - *Scale*
    - *Shear*

GPI Function Classes

- The GPI provides functions for:
  - Saving the data
    - *Metafiles*
    - *Retained graphical segments*
  - Manipulating text and fonts
  - Printing the picture
  - Managing presentation spaces

Program construction

- Use at least two threads
  - Main thread processes and dispatches messages
  - Second thread does the drawing
  - Use more threads (i.e. for lengthy calculations) as required
  - Keeps the application responsive to the user

Presentation Space

- Logically where the application draws
- 3 basic types:
  - Micro cached (WinGetPS)
  - Micro (GpiCreatePS - GPIT_MICRO)
  - Full (GpiCreatePS - GPIT_NORMAL)
- Very close mapping to a device context

Presentation Space

- Offers a variety of PAGE UNITS
  - PU_PELS
  - PU_LOMETRIC (0.1mm)
  - PU_HIMETRIC (0.01mm)
  - PU_LOENGLISH (0.01 inches)
  - PU_HIENGLISH (0.001 inches)
  - PU_TWIPS (1/1440 inches)
  - PU_ARBITRARY (matches device)
- Example: TEST1.EXE

Introducing Some GPI Primitives

- A set of GPI concepts with examples
- Concentrating on:
  - Drawing widelines, full arcs, partial arcs
  - Drawing polygons
  - Coordinate spaces and transforms
  - Generating a complex clip path
  - Using palette manager
  - Drawing a 256-color bitmap
  - Performance

Introducing Some GPI Primitives

- Basic attributes (some examples)
  - GpiSetColor(hps, CLR_RED)
  - GpiSetBackColor(hps, CLR_BLUE)
  - GpiSetBackMix(hps, BM_OVERPAINT)
  - GpiSetCurrentPosition(hps, pptlPoint)
  - GpiSetLineType(hps, LINETYPE_SOLID)
  - GpiSetMix(hps, FM_OVERPAINT)
  - GpiSetPattern(hps, PATSYM_HALFTONE)

Introducing Some GPI Primitives

- Simple lines
  - GpiLine(hps, ppoint)
  - GpiPolyLine(hps, lCount, aptlPoints)
  - GpiPolyLineDisjoint(hps, lCount,aptlPoints)
- Draws one or more lines, starting at the current position or from the first coordinate (in the case of disjoint lines)

Introducing Some GPI Primitives

- Wider lines
  - GpiSetLineWidth(hps, LINEWIDTH_THICK)
  - GpiSetLineWidthGeom(hps, lWidth)
  - GpiSetLineJoin(hps,LINEJOIN_ROUND)
  - GpiSetLineEnd(hps, LINEEND_ROUND)
  - Geometric line widths need to be used in conjunction with a path to generate a widened line

Introducing Some GPI Primitives

- Geometric wide lines
  - Generate using a path
  - Can specify line end style
  - Can specify line join style
  - Example: WIDELN.EXE

Introducing Some GPI Primitives

- Drawing Polygons
  - Use GpiPolygons
  - High speed polygon rendering
  - Draws one or more filled polygons
  - No need to generate using paths or areas
  - Polygons can be disjoint, i.e. not connected
  - Example: POLYGON.EXE

Introducing Some GPI Primitives

- **Full arcs**
  - Generate using GpiFullArc
  - Can adjust using GpiSetArcParams
  - Example: CIRCLE.EXE

Introducing Some GPI Primitives

- **Partial arcs**
  - Generate using GpiPartialArc
  - Can adjust using GpiSetArcParams
  - Example: PIE.EXE

Coordinate Spaces And Transforms

- **Various coordinate spaces are provided**
  - World
  - Model
  - Page
  - Device
- **Different transformations can be applied at different stages of the coordinate space pipeline**

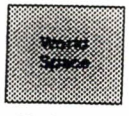
Coordinate Spaces And Transforms

- **Available transforms**
  - Model transform
    - *GpiSetModelTransformMatrix*
  - Viewing transform
    - *GpiSetViewingTransformMatrix*
  - Default viewing transform
    - *GpiSetDefaultViewMatrix*
  - Device transform
    - *GpiSetPageViewport*

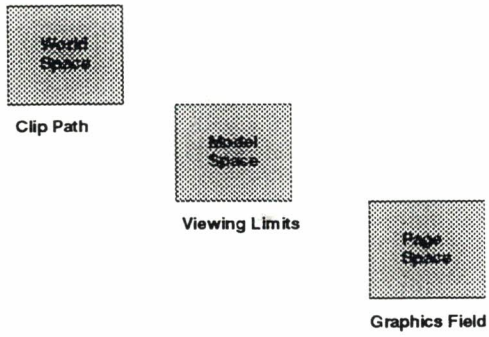
The Drawing "Pipeline"

The Drawing "Pipeline"

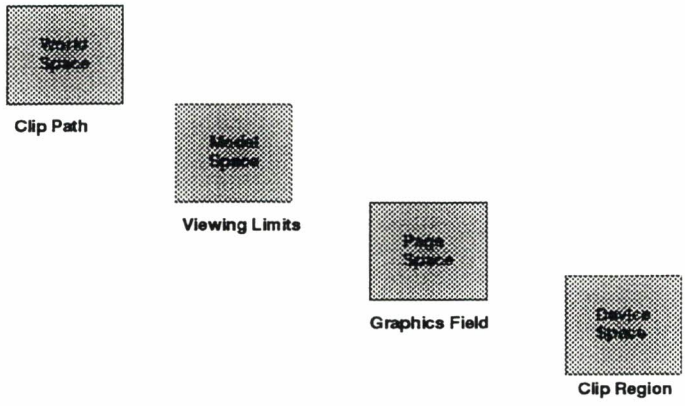
The Drawing "Pipeline"



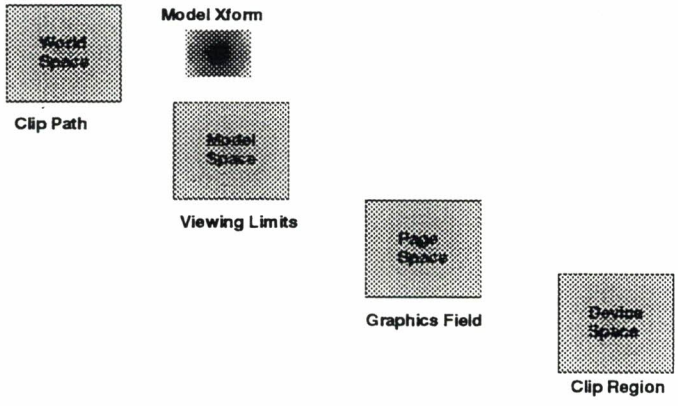
The Drawing "Pipeline"



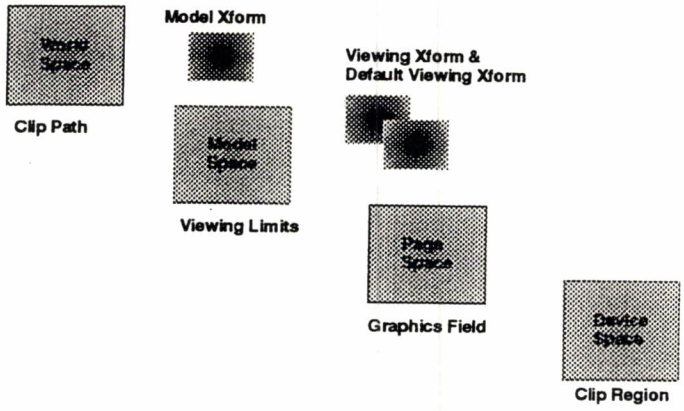
The Drawing "Pipeline"



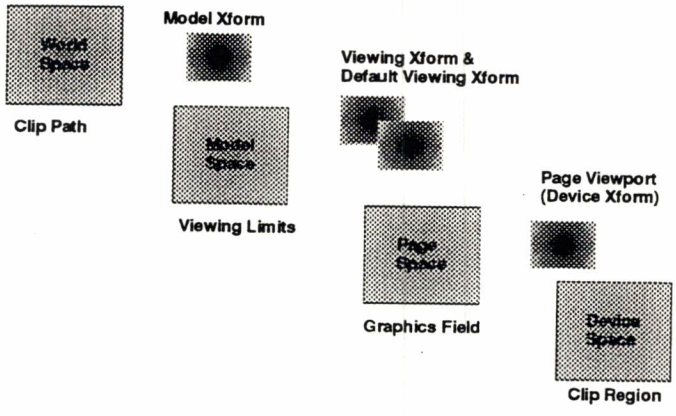
The Drawing "Pipeline"



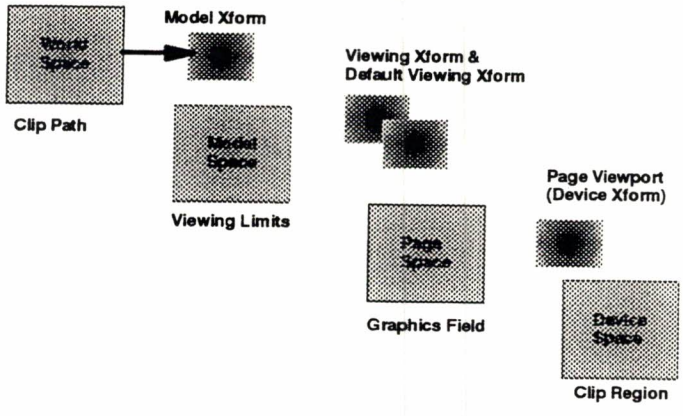
The Drawing "Pipeline"



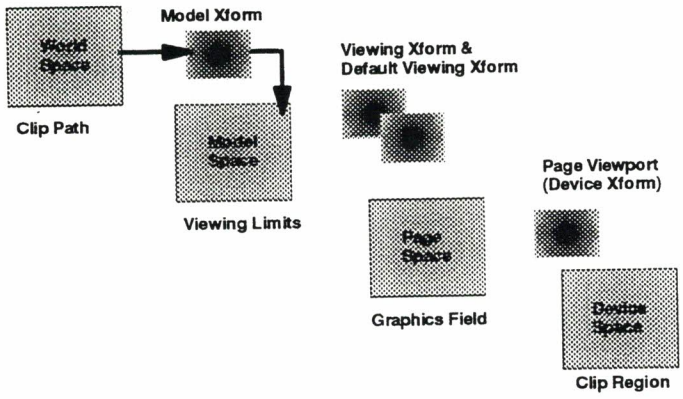
The Drawing "Pipeline"



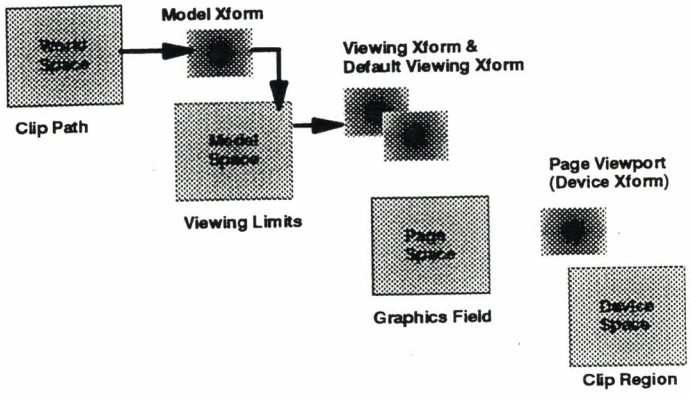
The Drawing "Pipeline"



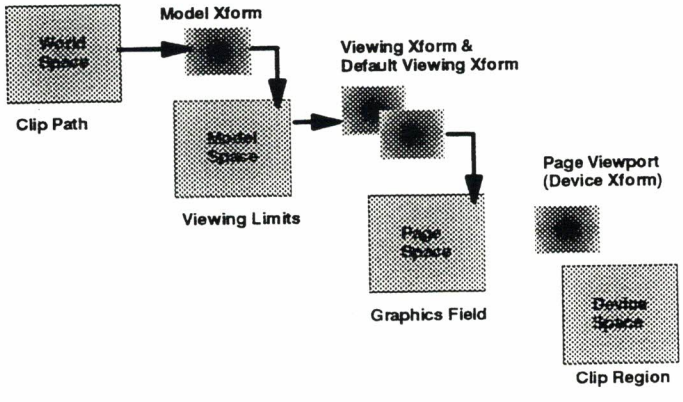
The Drawing "Pipeline"



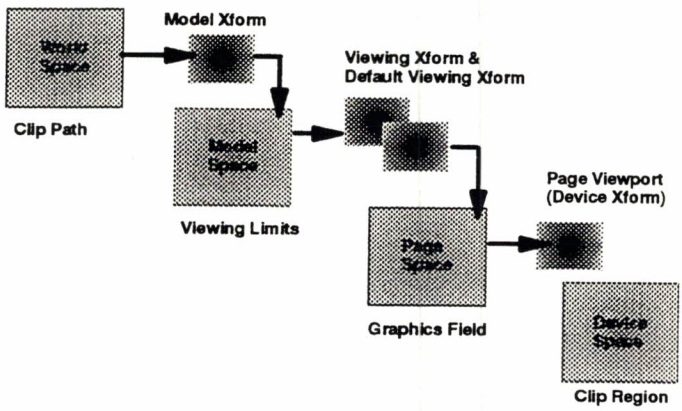
The Drawing "Pipeline"



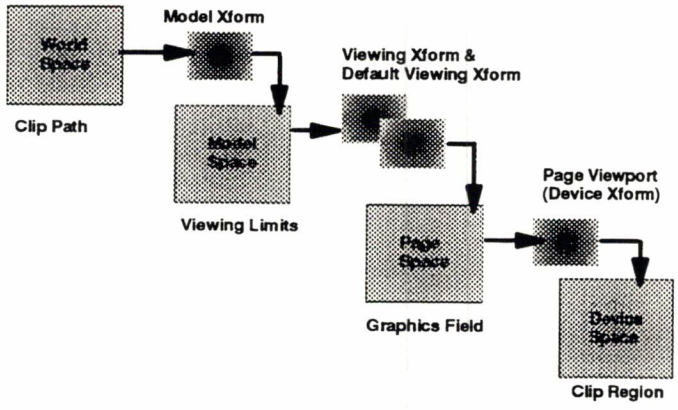
The Drawing "Pipeline"



The Drawing "Pipeline"



The Drawing "Pipeline"



Clipping Area Types

- **Clip path - world space**
  - Inclusive/inclusive, can include curves
- **Viewing limit - model space**
  - Inclusive/inclusive, rectangular
- **Graphics field - page space**
  - Inclusive/inclusive, rectangular
- **Clip region - device space**
  - Inclusive/exclusive, one or more rects.
  - Device dependent

Coordinate Spaces and Transforms

- The transformation matrix (MATRIXLF)

```
typedef struct _MATRIXLF {    /* matlf */
  FIXED fxM11;
  FIXED fxM12;
  LONG  lM13;
  FIXED fxM21;
  FIXED fxM22;
  LONG  lM23;
  LONG  lM31;
  LONG  lM32;
  LONG  lM33;
} MATRIXLF;
typedef MATRIXLF FAR *PMATRIXLF;
```

Coordinate Spaces and Transforms

- The transformation matrix (MATRIXLF)

$$\begin{bmatrix} fxM11 & fxM12 & 0 \\ fxM21 & fxM22 & 0 \\ lM31 & lM32 & 1 \end{bmatrix}$$

Coordinate Spaces and Transforms

- Transforming a point

$$\begin{bmatrix} X,Y,1 \end{bmatrix} \begin{bmatrix} fxM11 & fxM12 & 0 \\ fxM21 & fxM22 & 0 \\ lM31 & lM32 & 1 \end{bmatrix}$$

Coordinate Spaces and Transforms

- The transformation matrix (MATRIXLF)
  Scaling/Reflection use M11 & M12

  Rotation use M11, M12, M21, M22

  Translation use M31 & M32

  Shearing use M11 & M21 (vert)
  Shearing use M21 & M22 (horiz)

Coordinate Spaces and Transforms

- Note, if working in PELS, by default, all transforms start out as identity. Otherwise, the default transforms provide the mapping from world space to device space.

Coordinate Spaces and Transforms

- Side note: For speed, the graphics engine caches the following transforms

  - World to model
  - Viewing transform
  - Default (global) viewing transform
  - Page to device
  - World to device
  - Device to world
  - Plus, LRU of last two transforms not in above list

Coordinate Spaces and Transforms

- Three helper functions are provided for manipulating the transformation matrix:
  - GpiRotate
  - GpiScale
  - GpiTranslate

Coordinate Spaces and Transforms

- Translation
  - Adjusting position where object is drawn
  - Example: XFORM1.EXE

Coordinate Spaces and Transforms

- Rotation
  - Rotating object about a point
  - Example: XFORM2.EXE

Working with Fonts

- Bitmap vs. outline fonts
- Common font terminology
- Querying available fonts
- Creating logical fonts
- Character attributes
- Calculating spacing
- Outline font text as clip-paths
- Vector markers from an outline font

When Using Fonts, Plan Ahead...

- Decide up front if your application is a "screen only" application or if you are going to want to send output to other devices such as printers and plotters
- This will dictate the way you do your font handling in your application

Image Fonts and Outline Fonts

- With OS/2 are included a selection of image (bitmap) fonts and Adobe Type 1 (outline) fonts
- Other fonts can be added to the system dynamically

System Image Fonts

- Should only be used for "on screen" displaying of text
- Are not scalable (set of fixed point sizes)
- Their size is approximate for general display resolution
- Limited flexibility

Outline Fonts

- General use
  - Screen
  - Printer, etc
- Can be sized accurately with graphics
- Flexible uses
  - Outlines
  - Fills
  - Transformations (rotate, shear, etc.)
  - Clip paths, etc.

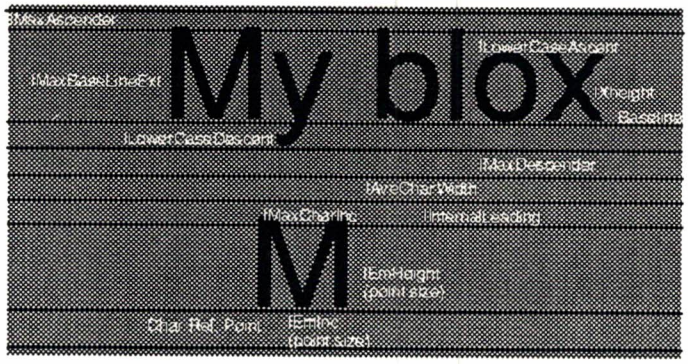
Outline Fonts

- Can achieve a proper transfer of what's on the screen to the printer

Device Fonts and Generic Fonts

- Device fonts are provided by a device or a device driver
  - lMatch value will be negative
  - Not available for use on other devices
- A Generic font is a system managed font that can be used on any device
  - lMatch value will be positive
  - Can be an image or an outline font

Some Terminology



The Installable Font Interface (IFI)

- The OS/2 graphics engine provides an installable font driver interface
- Font drivers for different font technologies can be written and added to the system dynamically
- An Adobe Type Manager font engine comes with OS/2 as standard

Querying Available Fonts

- Can use GpiQueryFonts
  - Private (QF_PRIVATE)
  - Public (QF_PUBLIC)
  - All (QF_PRIVATE | QF_PUBLIC)
- Can use system font dialog
  - Only shows image fonts that match the device font resolution

Offering the User a List of Fonts

- The system font dialog
  - Displayed using WinFontDlg(...)
  - Provides a uniform way to present lists of fonts to select from to the end user
  - Removes work from the application

WinFontDlg (Minimalist Approach)

```
FONTDLG fd ;
CHAR pszTitle[17] = "Font Dialog Test";
CHAR pszFamilyName[FACESIZE] = "\0";

memset( &fd, 0, sizeof( FONTDLG ));

fd.hpsScreen  = hpsClient;
fd.cbSize = sizeof( FONTDLG );
fd.pszFamilyname  =  pszFamilyName ,
fd.pszTitle  = pszTitle;
fd.usFamilyBufLen  = FACESIZE;
fd.fxPointSize = (FIXED)0;
fd.clrFore = CLR_BLACK;
fd.clrBack = CLR_WHITE;
fd.pszPreview = "Sample Text";
fd.fl = FNTS_CENTER;

WinFontDlg( hwndFrame, hwndFrame, &fd );
```

Using the Font Dialog

- Simple example: FONT1.EXE

Creating a Logical Font

- Query available fonts
- Select required font
- Call GpiCreateLogFont
- Select the character set as the current one
- Set character box size (if outline font)

- Example: FONT1.EXE

CreateLogFont

- Key things to provide in fattrs parameter
  - Size (of fattrs structure)
  - Facename
  - Codepage
    - *Non language font use 65400 - i.e., dingbats*
  - Image fonts
    - *lMaxBaselineExt, lAveCharWidth*
  - Outline fonts
    - *FATTRS_FONTUSE_OUTLINE*

Displaying the Text

- WinDrawText (display only)
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt

Character Attributes (Charbundle)

- Character set
- Mode
- Cell size
- Angle
- Shear
- Direction
- Text Alignment
- Extra
- Break extra
- Foreground color
- Background color
- Foreground mix
- Background mix

Character Set

- Set Using GpiSetCharSet
- Selects the current font

Character Mode

- Set Using GpiSetCharMode
- Mode 1 (CM_MODE1)
  - Ignores all attributes except direction
- Mode 2 (CM_MODE2)
  - Uses character shear, box, angle and direction, but the characters themselves are not sheared/rotated
- Mode 3 (CM_MODE3)
  - Obeys all attributes
  - Outline fonts only

Character Cell

- Set using GpiSetCharBox
- Used for sizing outline font characters
- Defines horizontal and vertical space occupied by an em square

Character Angle

- Set using GpiSetCharAngle
- Defines angle at which text is drawn
  - I.e., defines the angle of the baseline
- Actual appearance of the text is determined by the character mode

Character Shear

- Set using GpiSetCharShear
- Appearance of the text is dictated by the current character mode

Character Direction

- Set using GpiSetCharDirection
- Available directions
  - CHDIRN_DEFAULT
  - CHDIRN_LEFTRIGHT
  - CHDIRN_RIGHTLEFT
  - CHDIRN_TOPBOTTOM
  - CHDIRN_BOTTOMTOP

Character Alignment

- Set using GpiSetTextAlignment
- Available options
  - TA_NORMAL_HORIZ
  - TA_LEFT
  - TA_CENTER
  - TA_RIGHT
  - TA_STANDARD_HORIZ
  - TA_NORMAL_VERT
  - TA_TOP
  - TA_HALF
  - TA_BASE
  - TA_BOTTOM
  - TA_STANDARD_VERT

Character Extra

- Set using GpiSetCharExtra
- Defines extra space between character cells

Character Break Extra

- Set using GpiSetCharBreakExtra
- Defines extra space between words by increasing the size of the break character (usually the space)

Character Color and Mix

- Can use regular color/mix functions
  - GpiSetColor
  - GpiSetBackColor
  - GpiSetMix
  - GpiSetBackMix
- Can adjust just the character bundle
  - GpiSetAttrs (PRIM_CHAR)

Examples of Attributes at Work

- TEST1.EXE
- A program that demonstrates character manipulation

Calculating Line Spacing

- For image fonts
- Use lMaxBaselineExt to get a reasonable line spacing

Calculating Line Spacing

- For outline fonts
- Do NOT use lMaxBaselineExt
  - May be wildly inaccurate
  - Consider adding an integral sign to a font
- Good first approximation is 1.2 x point size
- However, depending on point size, line spacing should be chosen for a visually pleasing look

Outline Font Sizing - Device Coords

- Some general typesetting terminology
  - 12 points = 1 pica
  - 6 picas = 1 inch (actually 0.9961 inches)
  - 72 points = 1 inch
  - 1 inch = 2.54 cm
  - 1 inch = 0.0254 m

Outline Font Sizing - Device Coords

1. Choose point size required

2. Use following algorithm
   pt = selected point size
   dev = device graphics resolutions (pels/meter)
   mi = 0.0254 (meters/inch)
   ip = 0.9961/72 (inches/point)

   Set character box to : pt * dev * mi * ip

Fonts, Paths and Clipping

- Define a path
- Make clip path using GpiSetClipPath
  - Example: CLIPFNT.EXE

Fonts and Vector Markers

- Example of creating a logical font of vector markers and scaling them:
  - MARKER.EXE

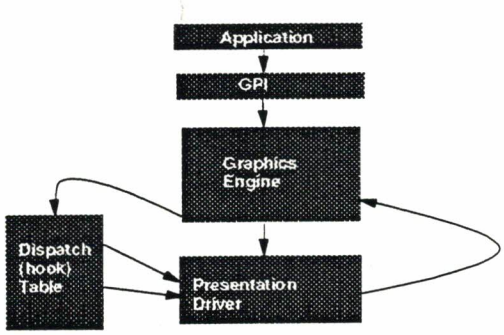
Presentation Parameters

- Quick and easy way to change the default font for a window or control
  - PP_FONTNAMESIZE
    - *e.g., 12.Helvetica*
  - Set using WinSetPresParam
  - Can be a little more exotic
    - *e.g., 12.Helvetica.bold.italic*

Palette Manager

- Example
  - Displaying the contents of the palette
  - How palette manager works
  - Example 1: PMPAL.EXE
  - Example 2: BITMAP32.EXE

The Graphics Dispatching Model



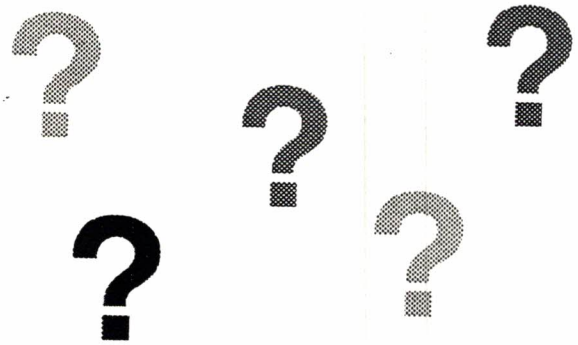
Performance

- Depends upon efficiency of
  - Application
  - Presentation driver
  - Hardware
- Can use large arrays of points if needed
- Understand underlying presentation driver model

And Finally...Putting It All Together

- Example application that uses
  - Disjoint polylines
  - Polygons
  - Animated palettes
  - Threads
  - Transforms
  - Performance concepts
- MOUNTAIN.EXE (A mountain generator) using subdivided triangles

Questions . . .



PRO 1107
Using the APPC Application Suite and Discovery

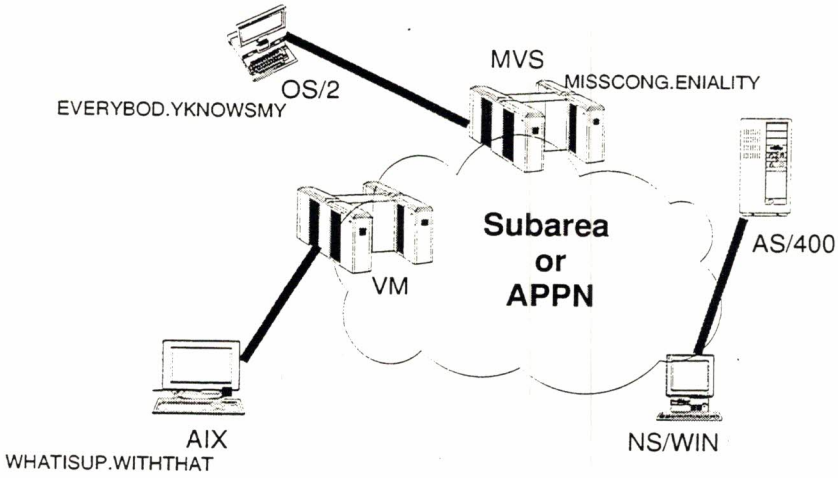
Dave Lutz
IBM

Agenda

- **Who:** YOU need these applications in your network
- **What:** APPC Application Suite and Discovery
  - AFTP, ANAME, A3270, APING, AREXEC, ATELL, Discovery
- **Why:** Common applications
  - Powerful network function
  - Standard across platforms/vendors
- **Where:** All IBM APPC platforms and soon from other vendors
- **When:** Now
- **How Much:** It's FREE!

APPC (C) IBM Corporation 1995 APPN

You have an SNA or APPN network: Now what?



APPC Application Suite Goal

- Reduce complexity for
  - End Users
  - Network Administrators
  - Application Developers
- By providing
  - Powerful applications that are widely available
  - Smooth migration from host-centric to client/server
  - Reduced configuration of APPC/APPN

APPC (C) IBM Corporation 1995 APPN

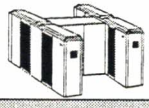
ANAME: Simple destination names

AIX

"*Cliff*"

OS/2

"*Norm*"


MVS

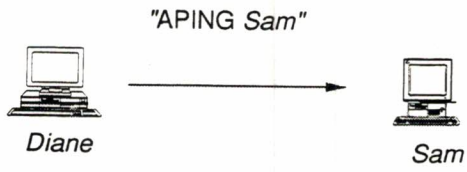
"*Carla*"

ANAME Server	
Cliff	WHATISUP.WITHTHAT
Norm	EVERYBOD.YKNOWSMY
Carla	MISSCONG.ENIALITY

- Easy-to-remember aliases
- ANAME server supports multi-platform clients
- API for easy use by applications
- ANAMELAD allows old applications to use ANAME nicknames

APPC (C) IBM Corporation 1995 APPN

APING: Can we talk?

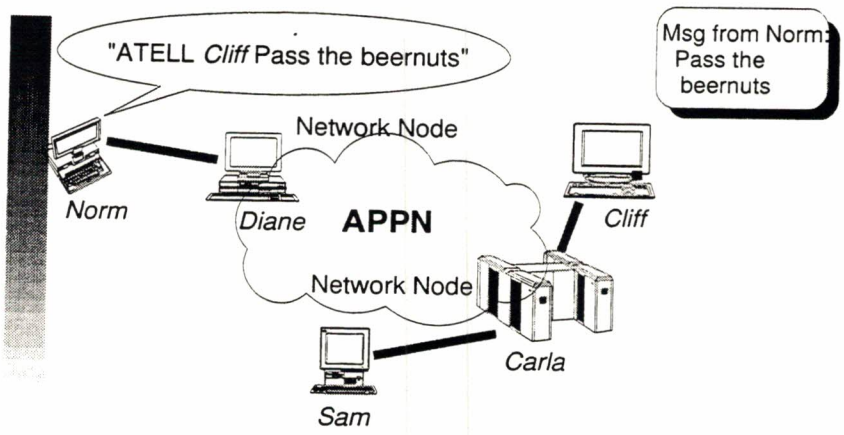


- Determines if APPC session can be established
- Displays detailed error information if session can not be established

★ **Very useful for problem determination!**

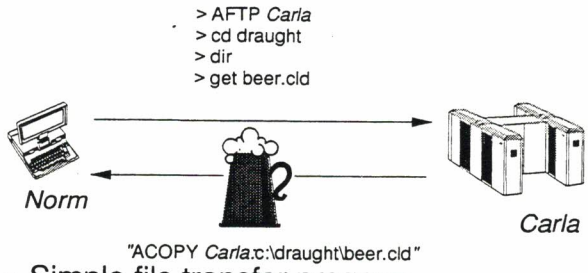
APPC (C) IBM Corporation 1995 APPN

ATELL: Send messages anywhere in your network



APPC (C) IBM Corporation 1995 APPN

AFTP: Transfer data quickly and easily

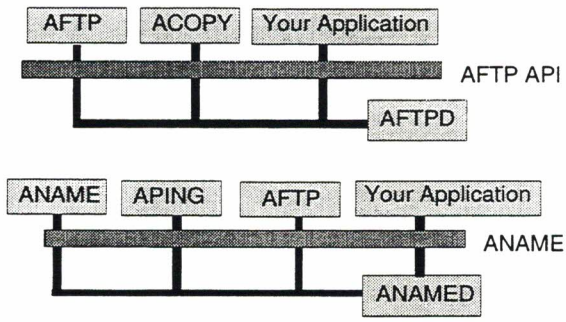


- Simple file transfer program
- Choice of interfaces
  - Interactive mode
  - Command line interface (ACOPY)
  - Application programming interface (API)

APPC (C) IBM Corporation 1995 APPN

Using the AFTP and ANAME APIs

★ *Your clients use our servers!*



APPN (C) IBM Corporation 1995 APPC

Why use the ANAME API

- Use ANAME in your application to:
  - improve application usability without extra configuration
  - provide ability to move partner application without changing input or configuration

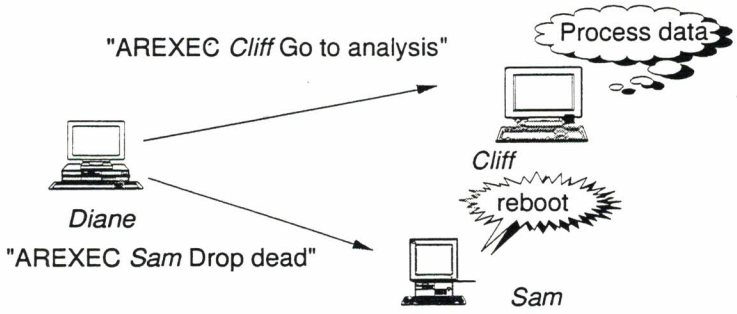
APPC (C) IBM Corporation 1995 APPN

Why use the AFTP API

- Use AFTP API instead of writing your own file transfer function to:
  - obtain multi-platform access
  - utilize fully tested and supported function
  - save development time and expense!
- Use AFTP API instead of ACOPY to:
  - use operating system resources efficiently
  - access complete file transfer functionality
  - define your own error logic
  - access user error messages

APPC (C) IBM Corporation 1995 APPN

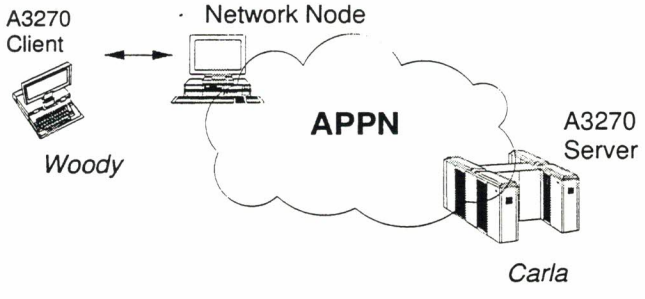
AREXEC: Power of remote operations



★ Authorized users can perform operations on any computer in their SNA or APPN network from a single computer

APPC (C) IBM Corporation 1995 APPN

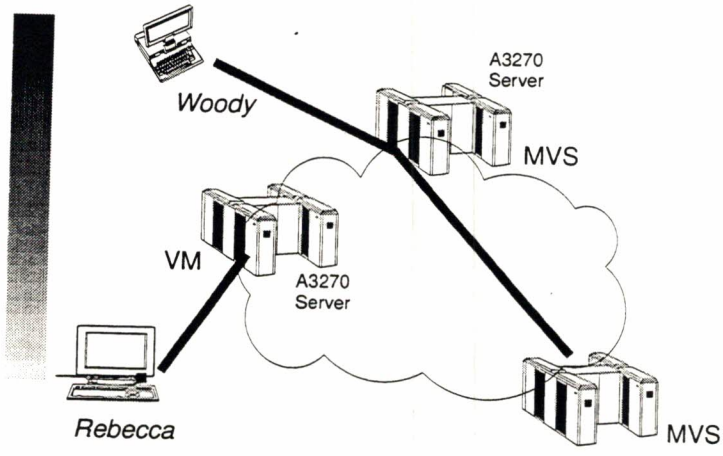
A3270: Smooth Migration to Client/Server



- Access and run existing host applications unchanged
- Allows APPN routing for 3270 applications

APPC (C) IBM Corporation 1995 APPN

A3270 in your Network



APPC (C) IBM Corporation 1995 APPN

Evolution of Link Configuration

Before APPN, SNA definition was a nightmare
- Link information required for EVERY partner

APPN has reduced the required definition
- End Nodes only need address of a Network Node
- Network Nodes need addresses of other NNs

Discovery makes it even simpler for LANs
- ENs and NNs need ***NO*** link definitions for APPN service!
- Low Entry Networking nodes need ***NO*** NN link definitions!

APPC (C) IBM Corporation 1995 APPN

Discovery is ...

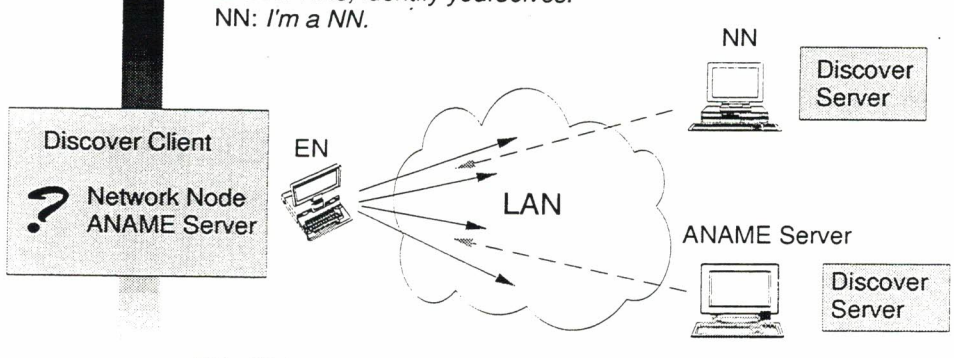
Dynamic LAN Address Resolution

- A mechanism for eliminating partner node definitions
- Primary use:
  - ENs and LENs finding NN servers
- Other potential uses:
  - Finding an APPC Name Server
  - Finding 3270 gateways
  - Network management, etc...

(C) IBM Corporation 1995

Discovery: Reduce configuration

EN: *All NNs, identify yourselves.*
NN: *I'm a NN.*



EN: *All ANAME Servers, identify yourselves.*
NS: *I'm an ANAME Server.*

(C) IBM Corporation 1995

Availability

- **OS/2**
  - AFTP, ANAME, APING, ATELL, and AREXEC
    - CM/2 1.11
    - Public repositories
    - APPC Showcase demo
  - A3270 client
    - Personal Communications/3270 V4 for OS/2
    - Public repositories
    - APPC Showcase demo
  - Discovery client and server
    - APPC Showcase demo
    - Next release of CM/2

APPC (C) IBM Corporation 1995 APPN

Availability (cont.)

- **AIX**
  - AFTP, ANAME, A3270 Client, APING, ATELL, and AREXEC
    - AIX SNA Server/6000 2.2
    - Desktop SNA for AIX
- **Windows**
  - AFTP, APING, and ATELL
    - APPC Networking Services for Windows
- **AS/400**
  - ATELL, APING and AREXEC
    - V3R1 QUSRTOOL
  - Discovery server
    - OS/400 V3.1

APPC (C) IBM Corporation 1995 APPN

Availability (cont.)

- **MVS and VM**
  - AFTP, A3270 Server, ANAME and APING
    - Optional feature with VTAM 4.2
    - Program product offering for VTAM pre- V4
  - AREXEC and ATELL
    - Public repositories
- **Vendors with source code**
  - Brixton, Microsoft, NetSoft, Wall Data

APPC (C) IBM Corporation 1995 APPN

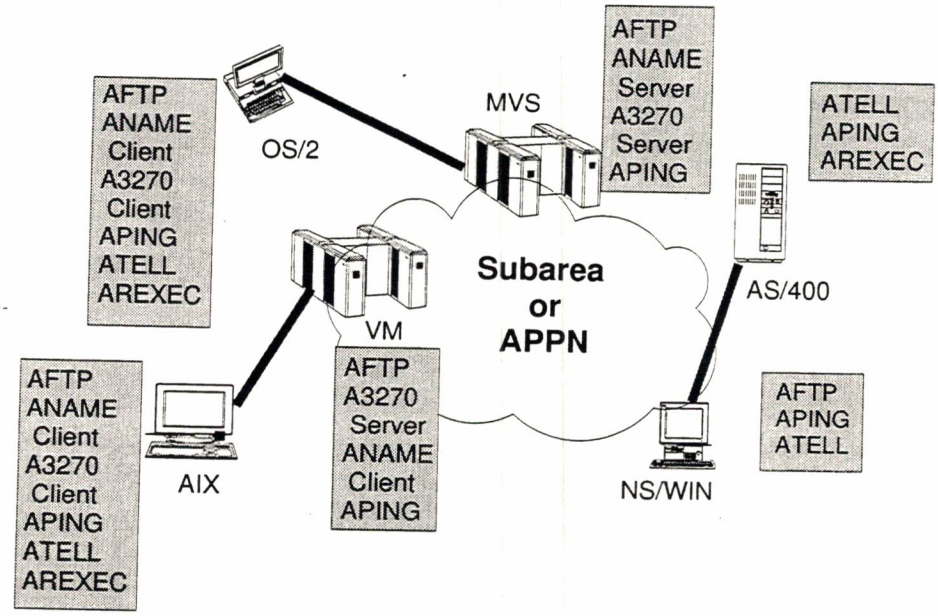
APPC Showcase

APPC - Quick and Easy!

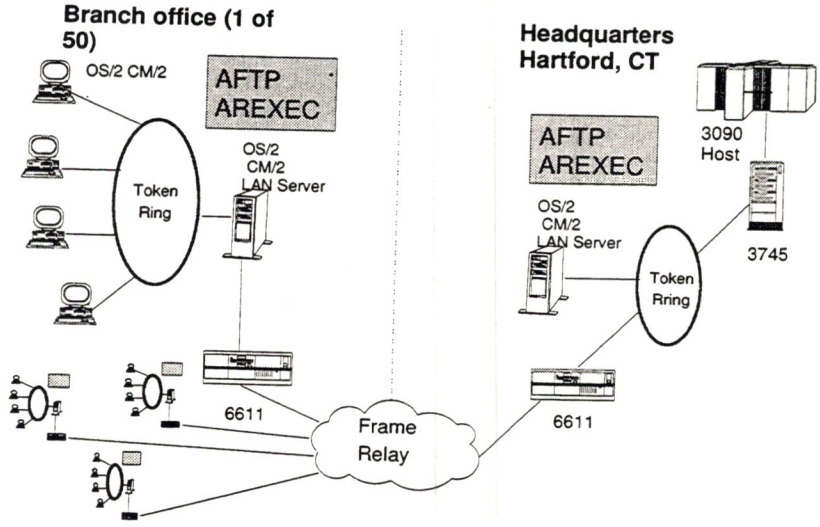
- AFTPPM - PM Interface to AFTP
- Discovery - find your network node automatically
- SNAPlay - PM Interface for all the applications
- Utilities and games
  - GETSENSE
  - NNLINK
  - Light Cycles
  - and more...

APPC (C) IBM Corporation 1995 APPN

APPC Application Suite in your Network



Konica Business Machines USA, Inc.



Summary

- **Common applications**
  - Modelled after industry standards
  - Identical interface across platforms
- **'Out of the box' availability**
  - Ships with CM/2, VTAM, AIX SNA Server/6000, Desktop SNA for AIX, Networking Services for Windows, and AS/400
- **Multivendor connectivity**
  - Source code licensed by 4 vendors and more expected
  - Provide specifications, some source code free of charge
  - Offer other source code at reasonable price

APPC (C) IBM Corporation 1995 APPN

Where to get the Showcase, Specifications, etc.

- WWW: http://www.raleigh.ibm.com/aac/aachome.htm
- FTP site: anonymous ftp to *networking.raleigh.ibm.com* (192.35.236.4)
- CompuServe: GO APPC
- OS2TOOLS (IBM internal)

APPC (C) IBM Corporation 1995 APPN

More information about APPC and APPN

- The Best of APPC, APPN and CPI-C CD-ROM (order number SK2T-2013)
- The APPC Connection
- The APPC/APPN Technical Conference World Tour '95
  - Chicago May 22-25
  - London September 4-7
  - Sydney October 16-19

APPC (C) IBM Corporation 1995 APPN

SYS 1101
Building Real-World Computer Benchmarks

Robert Chapman
Chapman & Associates
Aliso Viejo, Calif.

Welcome to the wonderful world of computer benchmarking. Computer benchmarking is used to determine the speed, capacity, and adequacy (general goodness) of a given computer. Computer benchmarking is used extensively by several different types of entities. The first, and most common entity is the computer press. The second is the computer industry. The third is the computer user.

The computer press uses benchmarks to allow them to differentiate similar products, so that subscribers and readers can determine "for themselves" which products give the most performance for the money. The computer industry uses benchmarks to highlight the basic fact that "their" product is better than "their competitors" product. The computer user has two major reasons for benchmarking. The first reason is determine if a given computer product can satisfy the users processing needs. The second reason is so that the computer user can "feel good" about their choice of equipment.

Theoretically, any consumer (note the use of the word consumer here, apparently this gives the impression that the people buying, using, and living around computers "eat" or transform these computers into waste. Why doesn't the government and everyone else simply use the word "people" or "person". Hereafter people will never be referred to as consumers in this article.) can review the public benchmarks associated with the computer products they are viewing and determine where in the scheme of computer processing they fit. This theory does not actually work in practice. First, all things are never equal in computing. Second, there are no general guidelines which can assist a person in determining the value of a given benchmark result. Third, many benchmarks have absolutely no value outside of the benchmarking environment, which means that they have no value even inside the benchmarking environment.

But first, it is important that people understand that there are numbers out there that are used in comparing products, which are not benchmarks, but should be treated as such. These numbers are "product specifications". Examples of these are processor clock speed and interface throughput speeds.

Most people love benchmarks, almost to the point of addiction. Many arguments (I mean discussions) take place over the meaning of the different benchmarks, and how they relate to specific equipment. Classic among these are (1) processor type and clock speed, (2) video performance based on Winmarks and etc., and (3) computer operating system throughput. And to be quite blunt, most of the benchmarks and the interpretations that people espouse are totally wrong or have no value at all.

This article is going to show you how not to succumb to the addiction of benchmarking, but instead be able to view a benchmark and immediately determine whether it has value, and if so what that value is. This article will open your eyes to the dangers of accepting common benchmarks. This article will also show you how to design simple, yet effective, benchmarks for yourself which actually do give you accurate information regarding computer processing capabilities.

FIRST, THE BASICS

All benchmarking starts by utilizing the following basic components - CPU (usually a microprocessor), bus, video adapter, I/O subsystem, applications, and finally operating system.

CPU's are usually identified by family, 8080, Z80, 6500, 6800, 80286, 68000, 80386, 68010, 80486, 68040, Pentium, PowerPC, ad infinitum. Within these families there are differences in clock speed, peripheral support, floating point number support, memory caching, and power management facilities.

One example of a CPU type is an Intel 80486DX executing at 33Mhz. This CPU can execute 33 million instructions in one second. But only if all of the instructions executing only take one machine cycle and they use only standard registers, or refer to data that is already resident in the 8KB cache integrated into the CPU. Unfortunately there are almost no applications available to the average commercial user which satisfy the above three constraints, so the average computer user can not get 33 million instructions executed in one second on an

Intel 80486DX 33Mhz CPU.

Bus types consist of ISA (Industry Standard Architecture, which by the way is not standard, and is not formally an architect), EISA (Extended Industry Standard Architecture, which is a standard and is architected as long as there are no ISA adapter cards installed in the computer), and MCA (Micro Channel Architecture, which is a standard and is architected). In additional to those three there are new buses appearing - the Local Bus, the VL Bus, and PCMCIA (the little credit card sized devices).

A bus is a strange thing. It consists of a series of wires (actually printed circuit board traces), and a set of technical specifications (both electrical and logical) which define how electrical signals can move from one end of it to another. The bus of early computers was actually a cable containing a large number of wires through which electricity was routed. Later implementations of the computer bus evolved into common signal path through which any number of computer peripherals communicated with the CPU.

Keep in mind that the bus is a pathway between peripherals and CPU where only one peripheral can be sending or receiving data at a time. This means that a bus can be a limiting factor on the throughput of a system. Currently the bus limits computer performance when memory and video adapters are connected to the CPU through it.

Video Adapters provide a direct window into the results of the processing. I once read that the human eye is capable of resolving a square matrix of 10,000 points horizontally by 10,000 points vertically at one meter distance. That is a grand total of 100,000,000 points of information. I also know from watching TV in Europe that I can perceive this information more than 25 times per second. In fact, many of my friends can not watch computer monitors that refresh the screen at less than 72 times per second. Because most peoples have the ability to perceive so much information so quickly the performance of video adapters is extremely important.

Another side to this need to present information quickly is the very real problem that many computers have trouble computing the data to be displayed, or the bus moving the data, fast enough to keep up with the video adapter. Therefore, video adapters are much more sensitive to other performance issues, than for instance, a disk drive.

Video adapters can easily be process data faster than the bus can deliver it, and in many cases faster than the CPU can generate it. The reason for this is simply the large amount of information that graphics images consist of. A standard VGA displaying 256 colors requires 307K bytes of data. If this information is to be totally updated 60 times a second the computer must create and the bus must transfer 18.42M bytes of data in one second. If you now move this to a super VGA adapter managing a 1024 by 768 pixel screen resolution with 72 refreshes per second and 256 colors you need 55.3 million bytes of data in a second. Extending this same resolution to 65 thousand colors or 16 million colors makes the numbers go to 110 million and 165 million bytes per second. If you ever wondered why it takes super computers to generate special effects for movies you now know the answer.

Compounding this problem of moving large amounts of data is the fact that some video adapters use specialized coprocessors instead of simply providing pixel mapped memory. Coprocessor based video adapters accept commands from the host such as "draw a line from position x1,y1 to x2,y2", "draw a circle with the origin at x1,y1 and a radius of 20", and "fill the area outlined by x1,y1;x2,y2;x3,y3;x4,y4 with pattern n". These coprocessor based video adapters decrease the load on the CPU and the bus significantly, thus allowing every other process in the computer system to execute faster.

I/O Subsystems determine both how much data can be associated with applications, and how fast that data will be made available to those same applications. I/O subsystems consist of the disk drives and controllers, parallel and serial ports. This discussion will be concerned with the disk and controller because they are the primary focus of most computer users. Most computers have one disk controller, and one to two disk drives. Most network server computers have more than two disk drives, but still only one controller.

Controllers have maximum data rates quoted. Disk drives have maximum data transfer rates, track to track access times, latency times, and average access times quoted. These are the product specifications and are sometimes generated by benchmarks, but often are computed from the component specifications. Quite often the data rates quoted for the I/O subsystems are totally unsustainable, and almost never achieved (see the section on disk benchmarks below).

Applications are the major key to performance. Applications range from transaction processing systems (such as a interactive business application), word processing - the most typical personal computer application, all the way to batch applications processing large amounts of information.

The applications you run, or develop for your computer, are the determining factor on computer performance. By this I mean that even though you can run a given application faster on a "faster" machine than on a "slower" machine, a simple change in the way the application executes may provide differences in performance that greatly over shadow any differences achieved through changing computers.

Also, another aspect that is important to remember is what type of applications it is that you run. I/O intensive applications may not execute any faster on a super fast processor than on a slow processor. CPU intensive applications (those which do almost no I/O) benefit from faster processors, but are not impacted by moving to a faster hard disk (but then very few applications are impacted by moving to a faster hard disk).

Operating Systems define the rules under which applications execute. These rules encompass such things as how the hardware is accessed, how much memory an application can access, how much and what type of hard disk spaces is available, and how many things an application can do at once. Operating systems provide the limits under which all applications operate, and these limits have a major impact on performance.

An operating system such as DOS supports only one executing program at a time. An execution environment such as Windows 3.1, which is based on DOS, allows more than one program at a time to be active, but only one program at a time to execute. Windows 3.1 round robins through the different applications after each application instructs Windows that it is ready to let some other program execute. Both DOS and Windows 3.1 applications do nothing when an application requests data from the I/O subsystem. In the army they call this the "hurry up and wait" syndrome.

OS/2 executes more than one program at a time, and when one application requests data from the I/O subsystem OS/2 lets the next program in line execute. This means several things, and you better put on your thinking caps for this discussion.

OS/2 provides more throughput than DOS or Windows 3.1. Throughput is a measure of how much work your computer can perform in a given amount of time. Throughput is a constant when only one application is involved. By this I mean that a single application which is the sole user of all of the computer resources has a set throughput. But when OS/2 runs multiple applications throughput goes up, however when Windows 3.1 runs multiple applications throughput goes down.

Here are the details. When multiple programs run at the same time there is a set amount of overhead involved in tracking them and switching between them. For the purposes of this discussion we can assume that there is the same amount of overhead under OS/2 as there is under Windows. This overhead goes up as the number of programs executing simultaneously goes up.

When a Windows program requests data from the I/O subsystem everything stops executing until the I/O subsystem is completed. So under Windows a program will execute, then request I/O services, then execute and then release the CPU back to Windows so that the next program can execute for awhile, and incur the task switching overhead discussed above..

When an OS/2 program requests data from the I/O subsystem OS/2 switches to the next program. So under OS/2

a program will execute, then request I/O services, then execute again, never releasing the CPU back to OS/2. Instead when the OS/2 program requests I/O services, instead of the CPU waiting for the I/O to complete OS/2 switches to the next program so it can run while the I/O is being performed. In this way the task switching overhead is not directly seen by the applications programs and the throughput of the system is improved.

You might be asking yourself, OK - but what has this got to do with benchmarking? The answer is that benchmarks designed for DOS and Windows do not measure throughput, they measure response time. So they will tell you something about the performance about OS/2, but they will almost never show OS/2 performing better than DOS or Windows. The problem is that this is incorrect. OS/2 does more work "in a given amount of time" than DOS and Windows, but it generally does not do any one thing faster.

It also means that applications designed around the DOS and Windows architecture will execute at a set speed, and OS/2 will not perform miracles and make them run faster. Applications which are designed around the OS/2 architecture can run faster than their DOS and Windows counterparts. But they must be designed by people who understand OS/2 features and facilities, otherwise they will run just like DOS and Windows programs.

The operating system design also plays a fundamental role in determining what applications can be developed. Most of the applications running under the DOS and Windows environments are single user applications. They work only with the keyboard operator. Even such circumstances as connecting to LAN's becomes tedious and unreliable.

Under OS/2 it is now possible to create new types of applications because the rules have changed. Reliability has improved and basic facilities have been extended. One of the most important rules that an operating system enforces relates to the amount of memory that any one application can access, and how it can be accessed. OS/2 for instance, allows each application to have up to 500 MB of memory, thus assuring that most applications imaginable today can be supported. OS/2 also provides built in task switching, so applications no longer have to be aware of what state they are executing in, and this simplifies program design, coding, and testing.

SECOND, THE TOOLS

Most of the visible characteristics of a benchmark can be determined through the use of a stop watch and a lot of brain power. By carefully considering what is being tested it is possible to come up with a test that will reflect the resource utilization of a single component. However, there are easier ways that the acquisition of benchmark information can be performed.

The advent of OS/2 has also been accompanied by a series of tools which allow the various subsystems of OS/2 to be monitored during program execution. Chief among these tools are Pegasus by C.O.L. Systems, its newer release, Osrm/2, also from C.O.L.Systems, and System Performance Monitor/2 from IBM. A less industrial strength product, but still very useful, is CPU Monitor Plus from Bon Ami Software.

These three products provide CPU times, File I/O counts, and Memory usage statistics. This is the visibility needed to analyze the executioin characteristics of a program or system..

BENCHMARKING INDIVIDUAL COMPONENTS

Benchmarking individual components is the simplest way to gauge performance. It is also, I am sorry to say, worthless. I will cover the reasons for this statement later. But, since almost every benchmark you see published today is based on individual components I better discuss it.

When constructing or viewing a benchmark on an individual component the single most important fact to keep in mind is that the benchmark must not allow other components to impact the results. A simple example is that

when testing a CPU one must not perform disk I/O because disk I/O is not related to CPU performance. A complication of this simple requirement though shows up is the reverse example of testing disk I/O. It is very difficult to test disk I/O without involving the CPU.

Another important criteria is that the benchmarks must be sized such that the overhead associated with starting and stopping the benchmark can be identified and removed from the benchmark results. This is accomplished by executing the same benchmark process for at least two different sample sizes. The results of the smaller sample size are subtracted from the results of the larger sample size, thus giving the results associated with processing the difference between the two samples.

Most importantly, the benchmark should be designed to approximate the type of processing that the target user will be executing. Another way of saying this is that the benchmark should approximate the users execution environment. Let us face the facts, a benchmark that models computing the physical characteristics of the inside of a star (kilometer by kilometer) is not going to show how a computer will perform on basic word processing or financial transactions.

THE CPU

Benchmarking techniques for CPU are split into two groups. One group benchmarks performance on CPU's that have internal or external memory caches. The second group benchmarks non cached CPU's.

Memory caching dramatically changes the performance results of a CPU. Memory caching also is perfect for generating totally unrealistic benchmark results. Any benchmark developed for a CPU must be designed with the knowledge of whether a cache is to be present or not. This is because memory caches which are large enough to contain all, or a significant portion, of the instructions and data being processing in the benchmark will execute substantially faster than when the instructions and data can not be kept in the cache.

Benchmarks for CPU's which support memory caching have to be divided into two types. The first type will generate values pertaining to sustainable processing performance. The second will generate values pertaining to maximum short term (otherwise known as burst mode) performance values. Benchmarks for CPU's which do not support memory caching typically generate sustainable processing performance values.

A CPU benchmark program must start by determining the mix of instructions executed by a typical application. This mix consists of the number of "load register", "store register", "move from storage to storage", "add", "subtract", "branch or loop", etc. instructions typically executed. In addition to the instruction mix there should also be an analysis of the mix of data types. This mix consists of byte data, two byte data, four byte data, numbers, bit fields, and character strings. In a caching CPU there is also a need to determine the average size application, along with the average size processing loop.

Once all of this information is available a benchmark can be devised and executed (with differing amounts of processing so that the overhead can be identified and discounted). Notice though that this benchmark is only applicable to one type of application. The benchmark can not be extrapolated to different applications because the instruction mix and data will change.

A general purpose benchmark can not be created because the instruction mix and size will be wrong for everyone. Even as I say this I must admit that there are a lot of benchmarks in the market place. Unfortunately even the most standard can be very misleading.

Landmark Research International Corporation sells the "Landmark" benchmark utilities. These programs test the CPU, math coprocessor, and video performance and give a general indication of system performance. To quote the help information from this product "CPU test forces the processor to show the effect of the instruction cache" shows you just how easy it can be to be mislead by a standard benchmark. When I execute this bench-

mark against the AMD 386sx 25Mhz upgrade for my IBM PC/AT model 339 it says that my computer is executing at the speed of a 32.66 Mhz AT instead of the 8 Mhz that it originally was. But when I run real life applications they do not run any faster under the AMD 386sx chip than under the Intel 80286. The reason is that the applications can not fit in the cache, and so most of my memory references need to go to the system board and memory card, each of which are executing at 8 Mhz.

Many people will say that a standard benchmark is of value for comparison shopping. To this I say that any one believing this should not bother benchmarking. It is much simpler, and just as accurate, to assume that the newer processor with the bigger number (8086, 80286, 80386, 80486, etc.) runs faster. If you are going to depend on standard benchmarks you should just buy the biggest processor.

But if you want to get the most value for your money you must benchmark, and you must be sure that the benchmark models your processing environment, not someone elses.

The Bus

There are no benchmarks generally available for the bus. To some the bus is treated as a simple wire which can be ignored for almost all processing needs. To others the bus is an integrated data movement architecture with performance limiting attributes. But the real reason that there are no benchmarks generally available for the bus is that these would be too complicated to create and analyze for results.

The throughput of a bus is based almost entirely on factors beyond the control of the bus. Memory speed determines the bandwidth of the memory subsystem. Disk controller speed determines how fast the data can move from disks to the memory, and from there to the CPU.

The bottom line is that most existing bus technology is plenty fast compared to the demands placed on it by the CPU, memory and I/O subsystems. It is true that there are some applications appearing today which are stretching the capacity of the ISA bus (such as LAN's encompassing 20 or more workstations), but for the most part, bus capacity is not an issue in computer performance.

That said, the type of operating system does determine how quickly a bus impacts performance. In an operating system such as OS/2 2.x I have seen evidence that MCA computers perform I/O faster than EISA and ISA computer systems.

The Video Adapter

Video Adapters have perhaps the most checkered past. Many magazines feature benchmarks which try to indicate video adapter performance. These have typically concentrated on timing how long it takes to display a given set of graphics data displayed on a monitor. These measurements typically come in two forms. The first indicates the number of data bytes written to the adapter in a second. The second indicates the number of Microsoft Windows writes that can be performed in a second.

Unfortunately, there have been vendors who have tailored their video adapters specifically to the published benchmarks, thus generating astounding values. Two areas in which this has occurred were the special case processing allocated to the recognition of a specific character string, and the buffering of drawing commands so that drawing objects which are not visible are ignored. The first area of tailoring was a great (short term) marketing move, it obviously took advantage of a terrible benchmark design. The second area, buffering data, was a technological advance that was probably prematurely condemned.

These examples highlight another problem with general purpose, industry standard benchmarks. Manufacturers may optimize the performance of their products based on the structure of a common benchmark. From a business standpoint this makes a lot of sense. It allows a product to perform exceedingly well in the product comparisons, and thus sell better or at a higher price. It is perfectly legal, and if the benchmark actually does model real life it

is my opinion that it is not only perfectly ethical, but a smart thing to do. If the benchmark does not model real life then the benchmark itself should not be in common use.

One last problem with benchmarking video adapters has to do with separating out the CPU and bus components from the adapter performance itself. Earlier in this article I discussed the amount of data necessary to display VGA and super VGA images. When benchmarks are dealing with high density (lots of colors) high resolution (lots of pixels) images the CPU plays a big part in creating the data. This has been seen to great advantage in the past when video adapter manufacturers come out with new versions of their video drivers and the video performance doubles. Or when the images saturate the bus when the images are memory mapped, but execute fine when the video adapter is coprocessed, and the video driver is aware of the coprocessor.

The I/O Subsystem

Benchmarking the I/O subsystem typically deals with the disk subsystem and consists of creating and reading files over and over again on the hard disk. In addition the benchmark must take into account the physical speed of the controller and the hard disk. This seems simple enough on the surface, but is fraught with danger.

There are two different file systems available today, FAT and HPFS. Each disk or disk partition is associated with one of these two file systems. FAT disks have a simple structure, easy to program for, easy to place data onto. HPFS is a more complicated file structure, harder to program for, harder to place data onto. FAT disks allow files to be spread across the disk, split up into clusters which can be located randomly around the disk. HPFS disks try to keep files physically close together to minimize disk head movement.

Benchmarks must be aware of the significant effects of these two philosophies. For instance, FAT disks will provide good performance on sequential access to small files, and large non-fragmented files. HPFS will provide good performance on sequential access small files and better performance on large files, and random accessed files. Neither FAT or HPFS will have much of a performance benefit over the other on non-fragmented disks. Neither FAT or HPFS will have much of a performance benefit over the other on heavily fragmented disks. However, an HPFS disk will fragment more slowly than a FAT disk.

Any benchmark which operates on newly loaded disks, or highly fragmented disks will not demonstrate the differences between FAT and HPFS.

Benchmarking which is not aware of the type of controller (caching or non-caching) has no value to the user. Many benchmarks used in the trade press disable disk caching when possible. The rational being that this levels the playing field. The problem with this is that the playing field should not be leveled. If someone pays more for a system, they want to get more out of it. And if a feature comes standard in a computer it should be part of the benchmark.

There are three types of tests which should be run on disk subsystems. The first is a burst mode test. This tells the user how quickly data can be passed under the best of circumstances. This is typically only a valid benchmark when caching controllers are available. The second is a sustained transfer mode test. This tells the user how fast data can be passed from a disk to the computer. This value is almost always related to three disk specifications, the number of bytes on a track, the amount of time it takes the head to move from one track to another, and finally how many times a minute the disk rotates (the level of fragmentation of the disk also has an impact). Notice that the speed of the controller and whether it has a cache or not does not come into play in this benchmark. The third test is the random read/rewrite test. This test indicates the maximum sustained performance a disk can support in a mixed application environment. Note that this test has only a small value for single-tasking operating systems like DOS and Microsoft Windows, but it has a large value for multi-tasking operating systems like OS/2.

OS/2 supports multiple applications executing simultaneously. These applications cause paging to occur at random intervals on the swapper.dat file, and also cause the reading and writing of data files at random intervals.

This means that even sequential file access being performed in an application can take on the attributes of a random read/write process because other tasks can be moving the read/write head of the disk around between any application's sequential I/O requests. Therefore, any I/O subsystem benchmark under OS/2 should be heavily weighted towards this type of processing.

The Applications

Benchmarking applications is the best of the stand-alone techniques. It allows a user to actually see the impact of different computer systems on their day to day environment. However, it suffers from the same problems as the CPU instruction mix mentioned above.

Even if two companies use the same application, they probably do not use it exactly the same way, and they definitely do not use it with the same equipment. And equipment makes a big difference in performance.

Many times applications are benchmarked against each other, three applications which are always being compared are the AMI, WordPerfect, and Word for Windows word processors. These benchmarks typically attempt to do a feature by feature comparison. They compare file load and save times. They compare spell checking and formatting times. They compare printing times. On the surface these benchmarks tell the casual user everything they need to know. The problem is that they don't tell the user everything that needs to be known.

In the above example it does not matter if one word processor loads and saves data faster than another if it is missing necessary features. Or worse yet, it may load and save data slower than the others if X memory is available, but faster than the others if Y memory is available.

In other words, there is usually too little data available from the benchmark testing to provide a truly good indication of performance and applicability for the typical user. Quite frankly, a user who uses only these types of benchmarks on which to base their buying decision is better off just buying the brand name word processor which is selling for the cheapest price at the time of purchase.

The Operating Systems

Benchmarking operating systems so that a performance comparison is possible is usually very difficult, and often impossible. Each operating system has different I/O subsystems, calling conventions, compilers, access methods, etc.. Fortunately for OS/2 2.x all of the DOS facilities are available, so it is possible to create benchmarks which come close to measuring performance of the two. But there are additional benefits associated with operating systems which can not be benchmarked by running identical programs.

OS/2 2.x can execute almost every DOS program known to man. There are exceptions, some companies explicitly write DOS programs that are incompatible with OS/2 2.x, so these can not be exactly compared. Even if OS/2 ran DOS programs faster than DOS, it still costs more to run them under OS/2 because of the minimum memory required by OS/2. On the other hand, programs written for native OS/2 may not have as many features as their DOS counterpart, but they may be much more reliable, execute faster, and allow OS/2 to service more applications simultaneously than if the DOS counterpart was executing.

Benchmarks developed for OS/2 should concentrate on providing indications of machine throughput, because an OS/2 machine is typically executing several applications simultaneously, even if the user does not realize it. Benchmarks developed for DOS (including Windows applications) should be designed to show maximum user response for single applications.

BENCHMARKING COMPLETE SYSTEMS

This is the only valid method for determining the performance and throughput of both the individual components and a complete system. This is because a computer system is more than the sum of the parts. It is a dynamic environment where every feature is interacting with the rest and this causes the performance characteristics to undergo continuous changes. In addition to the hardware considerations there is the application software and the operating system. DOS based computer systems are less dynamic than OS/2 based systems, but the ebb and flow of applications, along with the reading and writing of data on the hard disks and video subsystems still provides a constantly changing set of characteristics.

CPU benchmarking is dependent on the amount and type of memory cache, how large the working set (how many different memory locations are being accessed in a set amount of time) is, how many DMA devices are accessing memory, what the I/O interrupt rate is, and how large the video memory is. Bus benchmarking depends on the CPU speed, what DMA devices are executing, what type of adapter cards are placed in the computer, and whether the bus is allowed to be a master (initiate its own activities) or is always a slave. Video adapter performance is effected by the type of bus, the DMA facilities, the memory bandwidth, and whether the operating system or application must be able to read back the video adapters state and memory contents. I/O subsystem performance is based on the types of requests passed to it by the applications and operating system, and the load on the bus when the data is sent/returned. Applications have to be functionally equivalent and matched to the operating system. Operating systems must be tested on their features, not just on their ability of mimic a single applications.

When reading benchmark results keep in mind that the amount of physical memory available, the amount of cache in a disk controller, and the disk caching software have large impacts on overall system performance. Therefore, all results should be segregated based on the above three values. Never compare, for instance, the performance results of a system with 8MB of RAM with that of a system containing 4MB of RAM. Never compare, for example, a system with a 8KB internal cache with a system with an internal 8 KB cache and an external 128KB or 256KB cache. And never compare a system with a large caching controller with a system with no caching controller when dealing with small file sizes.

When looking at results from common published benchmarks remember that some system components may be tailored to the benchmarks. This means that the results you see may not be indicative of the performance you get on the system you use. Before you raise a cry about my being alarmist or insisting that this does not happen, you would be wise to accept a bit of cynicism into your outlook in this area. There are documented examples of video products being designed around specific benchmark characteristics. Many compiler companies use industry standard benchmarks to assist them in developing and judging the value of optimization techniques. In other words, benchmarks do not exist in a vacuum, and manufacturers are not total simpletons.

DOING YOUR OWN BENCHMARK

Even with all of the above in mind, benchmarks can be developed which tell the user which system is the best, and they are simple to construct. Any single user with an existing computer system need only perform a time study on their current work load. Using this information the user can request that the computer systems they are considering for purchase be set up with copies of their existing software. Then it is a simple matter to repeat some of the work and compare the time involved.

Benchmarking for new development, for example where a corporation is developing new software to distribute around the country is more difficult, but still possible. If the project team identifies the amount of I/O to be performed by the various applications, along with the frequency of each application, a small program could be written which could approximate that profile. Under OS/2 there are some difficulties with this approach that need

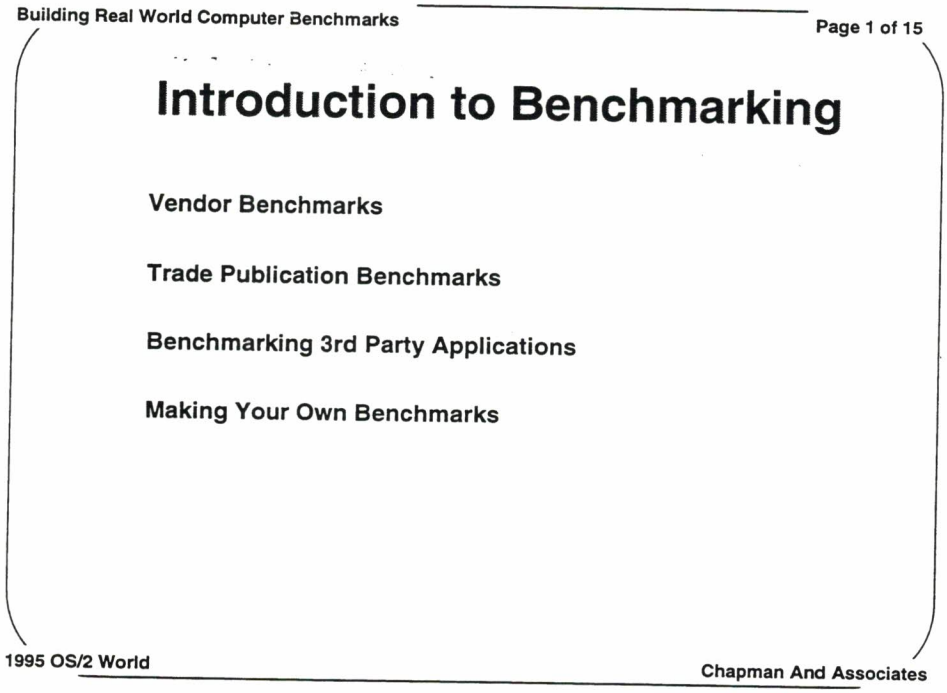
to be considered. First, the program should also attempt to approximate the memory usage of the real application. Second, the number of applications executing simultaneously should also be approximated. Once these are approximated you have a valid benchmarking system which can be used to determine relative performance among different platforms.

TO SUMMARIZE

The only thing complicated about benchmarking is deciding upon the applicability of the standard benchmarks published by the press and product vendors. Each benchmark must be scrutinized to determine whether it is appropriate to the analysis at hand. This is important to understand, especially considering the most of the published benchmarks are useless for gauging performance in the OS/2 environment.

Building benchmarks for a given environment is pretty simple. Existing application environments can be copied to the target computers and normal processing activity can be compared against the existing system.

Corporate development is slightly more complicated. The developers must anticipate how must I/O is performed, the frequency of the different types of applications, and the estimated memory requirements of those applications. Once this information is known it is a simple matter to write a program which models these parameters. This program can then be run on each of the target machines and a determination of which is the most appropriate can be made.



Vendor Benchmarks

Specifications

Industry Standards

Published Benchmark results

Truth In Advertising

Specifications

Disk Drives
- Maximum Transfer Rate
- RPM's
- Track Capacity
- Number of Read/Write Heads
- Simultaneous Head Access

Video Cards
- Memory Mapped
- Accelerators
- Memeory On Card
- Maximum Colors

Industry Standards

SpecMarks

DhryStones

Hobby Magazine Benchmarks

Published Benchmarks

Aggregated results

Suspicious Hardware Configurations

Incompatible Software Configurations

Nonsense Conclusions

Vendor Bias

Truth In Advertising

Some Products Are Designed Around Benchmarks

Benchmark Environments Are Always Manipulated For Maximum Marketing Value

Specifications Are Often Quoted As Benchmark Information

Differentiate Behind Hardware And Software Components In Benchmarks

Trade Publications

Good For Product Announcements

Good For Elementary Introductions To Technology

Historically Bad For -
- Perspective
- Detailed Investigations
- Technology Ramifications
- Vendor Bias

Gossip And Vendor Hype

Columnists, Pundits, and Gossip Mongers

Great Fun

Lose Sight of Why We Use Computers

***** We Want The Most Value For Our Money *****

Benchmarking 3rd Party Software

Itemize All Available Features

Ignore Each Feature You Will Never Use

Determine Which Packages Have Your Required Features

Identify Frequency Of Use Of Each Feature

Now Review Vendor/Journal Benchmarks

Create Your Own Benchmark For Features Not Covered or Distorted By Vendor and Journal Benchmarks

Make Your Own Benchmarks

Determine Your Operating Requirements

Determine Your Application Requirements

Decide Which Features Are Most Important To You

Structure Tests Around Your Requirements and Features

Do Not Aggregate Results

Determine Operating Requirements

How Much Disk Space Do You Need

What Display Resolution Is Required

Do You Know How Much Processor Speed Is Necessary

How Much Can You Spend

Are You On A Network - What Type

Determine Application Requirements

How Many Applications Will Execute Simultaneously

How Much Memory Does Each require (Working Set)

How Much Disk I/O and What Type (Read/Update/Add)

Which Functions Are The Most Important

Which Functions Are The Most Repetitive

Which Functions Cost The Most Money

Important Features

Rank Features By Importance
- Cost
- Value
- Criticality

Ignore Unimportant Features

Reevaluate Features Which Have No Value Or Cost Too Much

Structure Your Tests

Design Tests To Ask Questions - Not To Support Preconceived Opinions

Test Individual Functions First
- Disk I/O Rates
- Disk I/O Volumes
- Application Load/Remove Times
- Identified Functions
- Video Speed (If Necessary)

Combine Tests - Two At A Time, Three At A Time, ...

Do Not Aggregate Results

Aggregation Hides Contexts

Aggregation Misleads Management (They Think They Understand The Index Value And Draw Wrong Conclusions)

Aggregation Should Be Replaced By A Summary Of Results And Recommendations

Keep Personal Opinions Out Of Results - That Is The Scientific Way

SYS 1102
Keep 'em Running: How to Support End-Users

Esther Schindler
TGC
Caves Creek, Ariz.

What makes OS/2 support different?

- Hardware
- Software
- Wetware
- ...and the interfaces between them

Hardware downers

- The age-old driver problems
- Making a computer system a system.

Software downers

- The curse of flexibility
- Scarcity of application software that's ready for corporate prime time.
- Scarcity and invisibility of utility software for problem determination, recovery, etc.

People problems: Gaining user acceptance

- The Reluctant OS/2 User
- Who wants to be more efficient?
- "We don't have this happen in Windows."
- Even when they're willing, getting folks to exploit the capabilities of the OS.

People problems, continued

- Side effects of ignorance: A tendency towards "it must be OS/2's fault" from those who aren't committed to OS/2.
- Inertia, problems of changing from any one OS paradigm to another

Interfaces & Communication

- Internal corporate commitment to OS/2
- Fewer OS/2 resources
- The spirit of uncertainty... are we choosing the right path?
- The OS/2 ISVs are often from such small companies... are they viable?
- Finger-pointing, system interoperablity

Solutions: Hardware

- OS/2 forces you to buy better, more expensive brands
- Warp has better driver support
- Pre-loads

Solutions: Software flexibility

- Standardizing directory names
- Cmd prompt on WPS pop-up menu so you can have some things known
- Use REXX to lock a folder, make objects undeleteable
- Remove/limit flexibility using DeskMan/2 or other utilities
- Tools to let you see what the user did with/to her desktop.

Solutions: Software

- You guys have to ask for the applications -- you have the clout.
- Letters to the editor *work*.

Solutions: User Acceptance

- Bi-monthly user group meetings
  - Present new tips and tricks for the applications and the operating system. Special sessions, late afternoons, for people with questions or concerns.
- Newsletters, "tip of the week"
- Training to develop enthusiasm, turn them into self-selected users
- Real, formal training
  - Like the weather, everyone talks about it, but nobody (few) do anything about it

Solutions: Exploiting the OS

- Self-selected OS/2 users are apt to be more technically savvy and need little support. They're also less likely to follow IS's lead.
- Ease into OS/2 using Boot Manager and Dual Boot. (Spend an hour a day with OS/2)
- 6 month transition period
- Show them how to do stuff; it won't occur to them to look

Solutions: Exploiting the OS, continued

- Turn 'em loose and let them play. Give 'em a copy of Warp to take home and play with. Show copies of Barnes' videos, other demos that work as training vehicles.
- Those who use (good) OS/2 apps seem more likely to multitask and discover the benefits of the OS.

Solutions: Interfaces & Connectivity

- Pros and cons of remote boot
- Running applications from the server gives support folks more control over setup etc
- Tools to figure out what users have
- CID

Unresolved issues

- Have we chosen the right direction?
- The application situation
- Some things just take time

SYS 1104
Implementing the GSS-API in your Client/Server Application

Mark Davis
Ray Scanga
IBM

Programming the Generic Security Services API

Mark Davis
Ray Scanga

Purpose of GSS API

- **Provides Generic Security Interface**
- **Mechanism independent**
- **Security Context between two Principals**
  - Security Context: A secure connection
    - Authenticate one or both principals
    - Data integrity checking (strong)
    - Data confidentiality (U. S. export controls)
  - Principal: a person (user) or program
    - Examples: application server (transaction processing)
    - Also: network adapter card, router, directory service
- **Some "human readable" data**
  - Principal names
  - Error messages
- **Many opaque data types**
  - Application manipulates only pointer or handle

Main Interfaces

- **Establishing a Security Context**
  1. gss_init_sec_context
  2. gss_accept_sec_context
  3. gss_delete_sec_context
  4. gss_process_context_token
  5. gss_context_time
- **Maintaining Per-Message Security**
  1. gss_sign
  2. gss_verify
  3. gss_seal
  4. gss_unseal

Name Manipulation

- **Names**
  1. gss_import_name
  2. gss_compare_name
  3. gss_display_name
- **Managing Credentials**
  1. gss_acquire_cred
  2. gss_inquire_cred
  3. gss_release_cred

Copyright 1994 IBM Corporation

Utility Routines

- **Releasing Objects**
  1. gss_release_name
  2. gss_release_buffer
  3. gss_release_oid_set

- **Message Display**
  1. gss_display_status

- **Mechanisms**
  1. gss_indicate_mechs

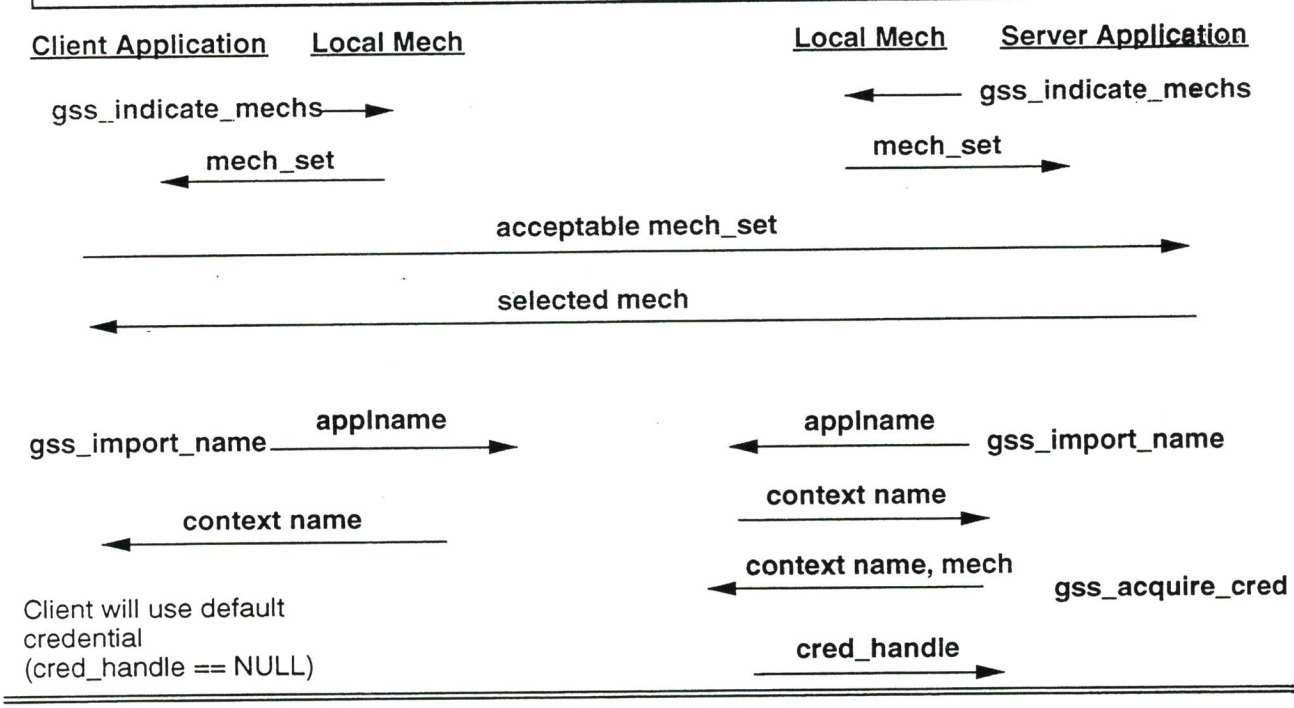
GSS API Framework

- **Objective:**
  - Dynamic support of multiple mechanisms
- **Parameters of Design:**
  - No change to existing GSS API
  - Registration process for mechanisms
  - Management API for registration/deregistration
  - Mechanisms support special GSS API entry points
- **Design in Progress**
- **Path to acceptance is a Draft RFC**

Copyright 1994 IBM Corporation

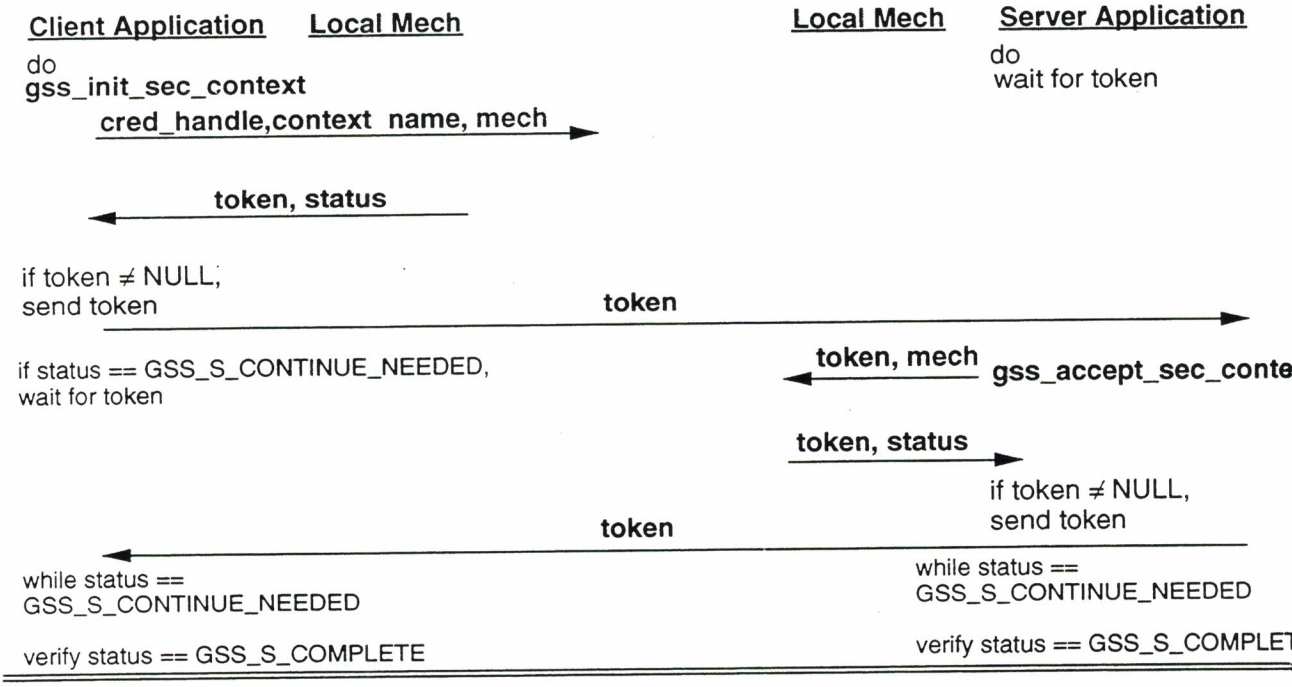
GSS Program Development

Select Mechanism/Manage Security Credentials



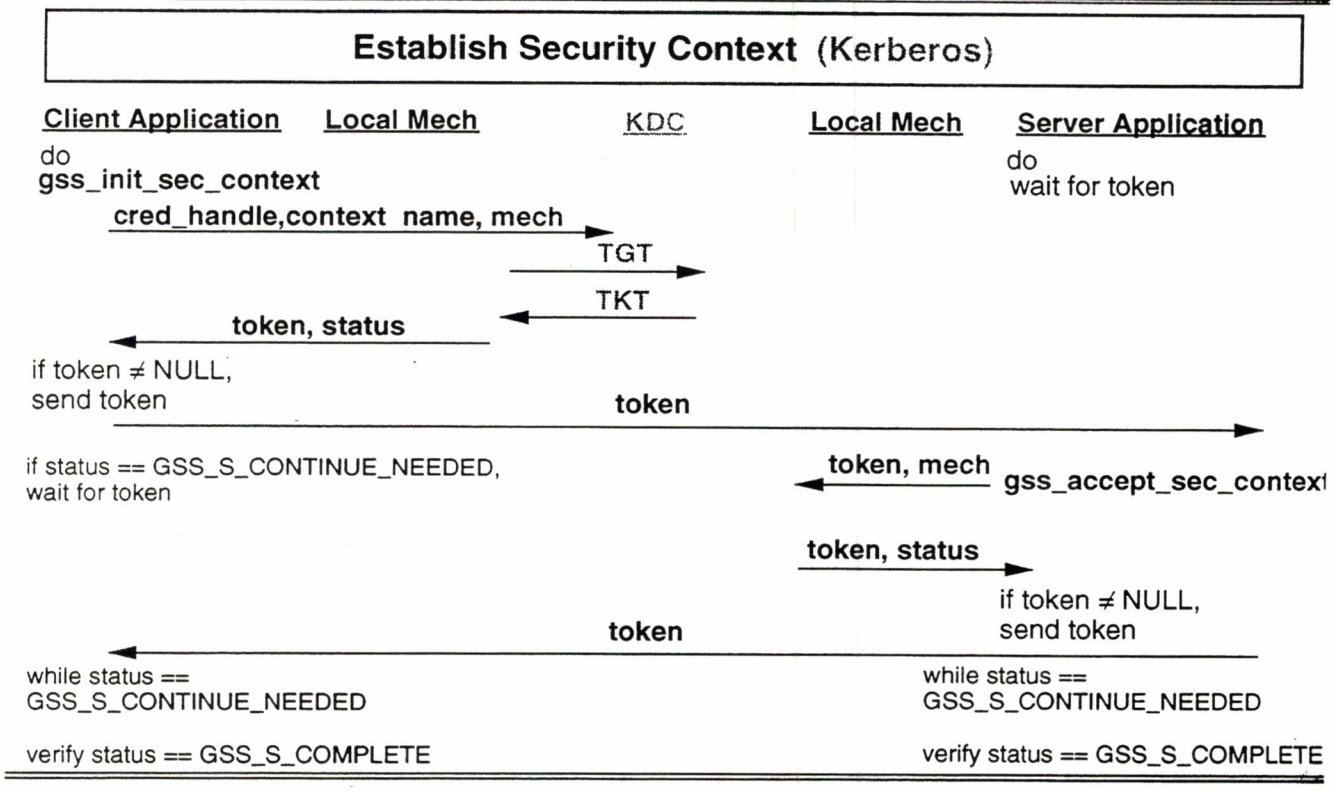
GSS Program Development

Establish Security Context

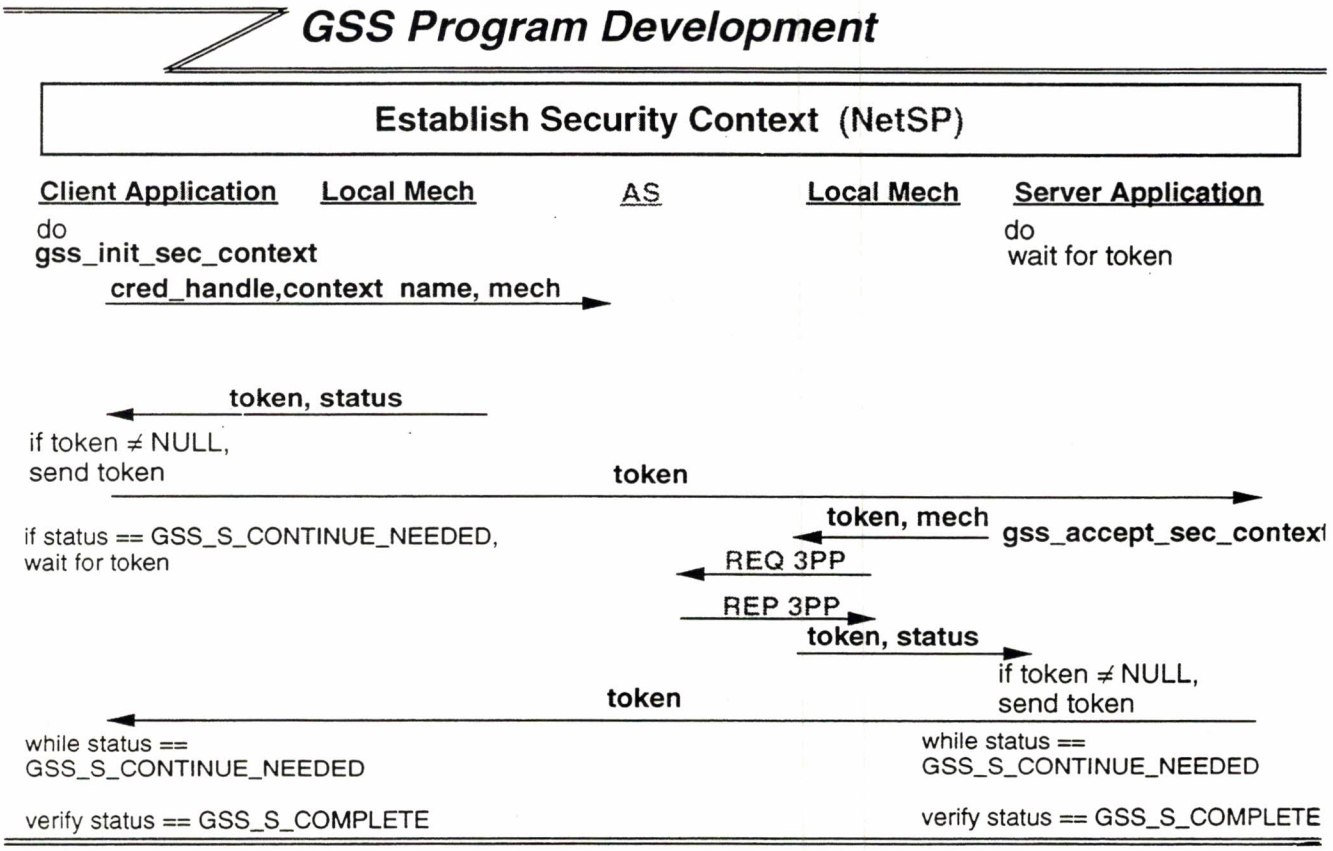


Copyright 1994 IBM Corporation

GSS Program Development



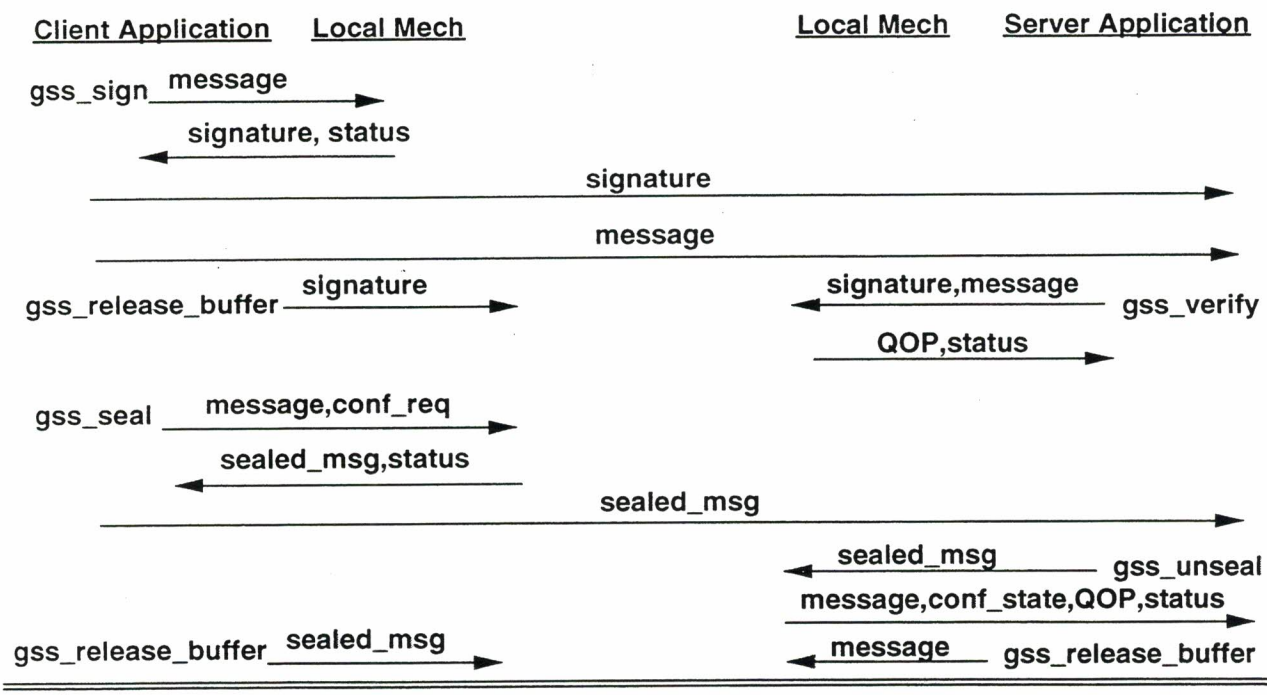
GSS Program Development



Copyright 1994 IBM Corporation

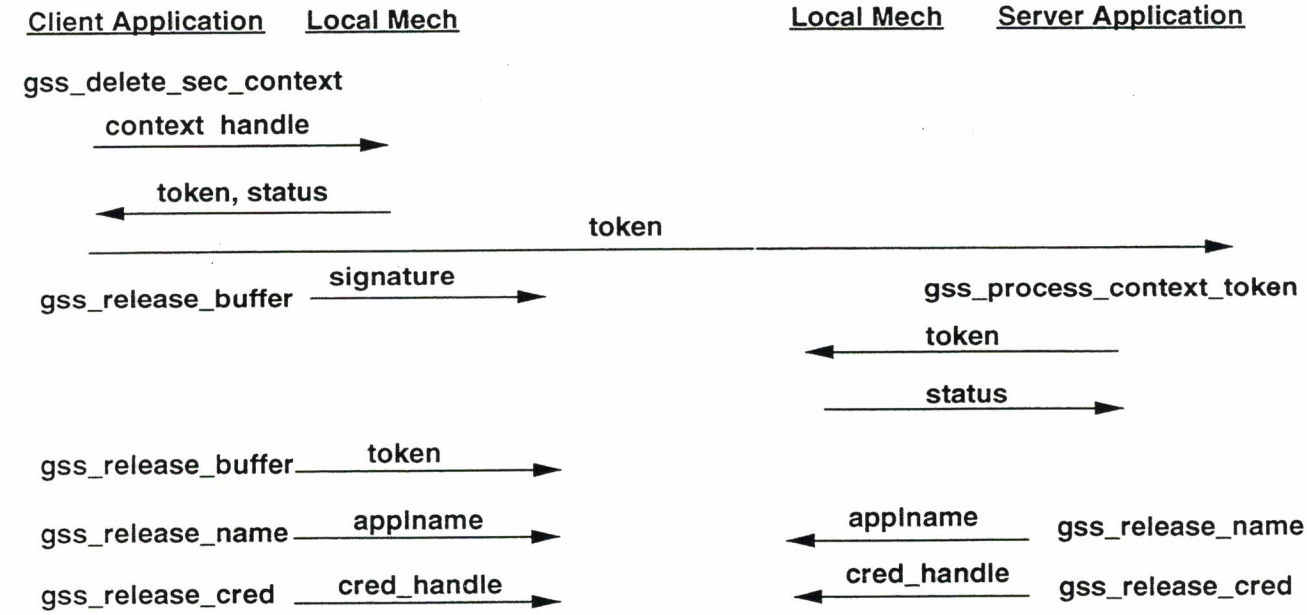
GSS Program Development

Information/Message Protection



GSS Program Development

Removal of Security Context and Credential



Copyright 1994 IBM Corporation

Development Process - Example Program

- **sample.c - Source code**

```
#include <gss.h>

main ( )
{
   OM_uint32              minor_status ;
   OM_uint32              status_value ;
   int                    status_type=GSS_C_GSS_CODE ;
   gss_OID                mech_type=NULL ;
   int                    message_context=0 ;
   gss_buffer_desc        status_string ;
status_value=GSS_S_NO_CRED;
gss_display_status (&minor_status, status_value,
                    status_type, mech_type,
                    &message_context,  &status_string);
printf  ("GSS Display: %s \ n" , status_string.value);
```

Development Process - Make Files

- **sample.mak - OS/2 Sample make file**

```
ALL:  sample.exe

sample.obj :  sample.c

cl  /c  /MD sample.c

sample.exe:  sample.obj
  link  sample.obj,  ,  , kk  llibcmt,  ,
```

- **sample.make - AIX Sample make file**

```
ALL:  sample

sample:  sample.c
   xlc  -o  sample  sample.c  -lkryp
```


Copyright 1994 IBM Corporation

SYS 1110

32-bit Migration Tips & Techniques

No paper available. Please contact speaker

this page intentionally left blank

CS 1209

Build an OS/2 Client/Server Application

No paper available. Please contact speaker

this page intentionally left blank

LAN 1202
Optimizing the Netware Client for OS/2

Bob Panza
The Systems Expert, Inc.
Woodbury, Conn.

SYNOPSIS

net.cfg options, settings, and parameters will be reviewed in detail for the latest version of the software (NetWare Client, v2.1 and OS/2, v2.11 at the time of this writing). Various implications of optimization will be discussed. This session is part of a series on the NetWare Client for OS/2 which includes #086, #093, and #101.

Agenda

1. Daemon configuration
2. Display hard errors
3. Link driver
4. Drivers, frames, and protocols
5. Link support
6. Source routing
7. ODINSUP
8. IPX
9. SPX
10. Novell NetBIOS
11. Named Pipes
12. NetWare Requester: Performance and security
13. NetWare for OS/2
14. Virtual MLID for LAN sharing
15. TCP/IP entries
16. Effects on multiprotocol interoperability

OPTIMIZING THE NETWARE CLIENT FOR OS/2

By design, parameters are the keys to precision tuning of network software. The result is a flexible system which can be readily adapted to meet different real world requirements. Survival of a product in today's network environment demands such changeability. Novell's NetWare Client v2.01 for OS/2 is a chameleon that fits the bill.

Through flexible architecture, the NetWare Client opens many doors of interoperability for OS/2-based network nodes. Support covers SPX/IPX, TCP/IP, NetBIOS, Named Pipes, LLC, ODI, NDIS, NetBEUI, IBM Extended Services, IBM LAN Services, multiple concurrent protocol stacks, DOS, MS-Windows, OS/2, Token-Ring, Ethernet, and so on. All are welcome to the party. The potential for interworking quickly becomes Napoleonic in scale, Bismarkian in unity, and Noordian in all encompassing connectivity.

Our tuning exploration will lead us to a few of many interoperability destinations. But there are numerous parameters to consider before venturing onto the high seas of interoperations. Within the NetWare Client, *net.cfg* is the file where tuning takes place. However, *config.sys* and *protocol.ini* soon enter into the picture for interworking.

NET.CFG

The sample *net.cfg* file illustrates all options in capital letters. Default value is shown at the end of every line in numeric or lower case characters. Each section contains a brief definition as ";" remark in lower case characters.

```
DAEMON CONFIGURATION  MESSAGE TIMEOUT no timeout if not present in net.cfg
;Controls the length of time network related pop-up and
;error related messages stay displayed on the screen.
```

DISPLAYHARDERRORS yes (hard error messages display) ;Displays hard errors and requests a user response.

```
LINK DRIVER driver name
```

```
DMA set by driver
FRAME set by driver
INT set by driver
MEM set by driver
NODE ADDRESS adapter address
PORT set by driver
PROTOCOL ipx, id and frame type are set by driver
SLOT number
;Specifies the LAN driver configuration for a single NIC.

LINK SUPPORT BUFFERS 20,1514 (20 buffers @ 1514 bytes each)
;Tunes size and number of requester communication buffers.

NAMED PIPES
CLIENT SESSIONS 16
SERVER SESSIONS 32
SERVICE THREADS 3
;Allocates resources to Named Pipes sessions.

NETWARE NETBIOS
ABORT TIMEOUT 30,000 milliseconds
INTERNET on
BROADCAST COUNT with internet on,  4 queries;
         with internet off, 2 queries
BROADCAST DELAY    with internet on,  2,000 milliseconds;
                   with internet off, 1,000 milliseconds

COMMANDS 32


LISTEN TIMEOUT 6,000 milliseconds
NAMES 24
RETRY COUNT 20
RETRY DELAY 500 milliseconds
SESSIONS 16
VERIFY TIMEOUT 3,000 milliseconds
;Allocates resources for NetBIOS names and sessions.

NETWARE REQUESTER
CACHE BUFFERS 8
DIRECTORY SERVICES OFF online not present in net.cfg.
LARGE INTERNET PACKETS OFF online not present in net.cfg.
NAME CONTEXT root of the tree
NONDED SERVER no sharing of NIC with NetWare for OS/2
PACKET BURST OFF on PREFERRED SERVER none
PREFERRED TREE root
REQUEST RETRIES 10
SESSIONS 8
SIGNATURE LEVEL 1
;Controls requests between workstation and file server.

PROTOCOL ODINSUP
    BIND driver of first NIC located
;Allows NDIS stacks to communicate with ODI drivers.

PROTOCOL STACK IPX
BIND 1st ODI driver in config.sys
ROUTER MEM 450 bytes
```

```
SOCKETS 64
;Tunes IPX communication between apps and drivers.

PROTOCOL STACK SPX
ABORT TIMEOUT 30,000 milliseconds
LISTEN TIMEOUT 6,000 milliseconds
RETRY COUNT 20
SEND TIMEOUT in milliseconds, determined dynamically by SPX
SESSIONS 16
VERIFY TIMEOUT 3,000 milliseconds
;Tunes SPX network connections.

NETWARE FOR OS/2
PERFORMANCE TUNING 7 (timeslicing, on a scale of 1 to 10)
REMOVE SERVER MEMORY (not released on DOWN)
SERVER MEMORY 8 megabytes (minimum)
;Sets NetWare for OS/2 memory and performance features.

PROTOCOL ROUTE
    SOURCE ROUTE (no def gbr or mbr) nodes 16 board 1
;Configures source routing broadcasts and table entries. The
;default is the absence of broadcast parms def, gbr, and mbr.

VIRTUAL MLID for LAN Sharing
    VIRTUAL BOARD SIZE 1514
;Tunes LANSHARE.SYS buffer size for NIC sharing between the
;NetWare Workstation for OS/2 and NetWare for OS/2.
```

PERENNIAL PACKETS

In the NetWare Client configuration file, net.cfg, parameters cover a lot of ground. Each option may contain multiple settings which are indented on lines following the option line. Important net.cfg options and settings are defined in this and following sections. Values of their variables are described in lower case.

- LINK DRIVER token
- DMA (not applicable to token)
- FRAME token-ring
- INT 2 •
- MEM d800
- NODE ADDRESS 400010000BAB
- PORT A24
- PROTOCOL ipx e0 token-ring
- SLOT 3

Driver name corresponds to a particular network board or interface. Drivers such as *token.sys* and NE2000.SYS respectively represent IBM Token-Ring and Novell NE2000 Ethernet adapters. In contrast, *odinsup.sys*and *lansup.sys*are drivers for interfacing with higher or lower layer software. *odinsup.sys*enables higher layer IBM Extended Services and LAN Services to use Novell ODI drivers as a means of accessing the network adapter. In a similar manner, *lansup.sys*interfaces between the NetWare Client and low layer drivers of IBM Extended Services that provide for adapter access. DMA, INT, MEM, and PORT are actual board settings. NODE ADDRESS allows the "burned-in" address of the adapter to be overridden with a user supplied address. SLOT represents the location of the board in the workstation node.

For purposes of our tuning journey, FRAME and PROTOCOL are the two most important settings within the LINK DRIVER option. FRAME essentially defines a type of packet at the data link layer. Each topology has a set of frame types. A particular protocol will utilize a particular frame type on a particular topology, thus establishing a protocol to frame type to adapter relationship.

Different packet types vary in format, typically in the data link header. The packet type definition represents a logical board within the contents of the physical board. There can be multiple logical boards initialized on one physical adapter board via multiple frame types. The physical board driver, e.g., *token.sys* or NE2000.SYS, or standard interface driver, *odinsup.sys*or *lansup.sys*, must be capable of supporting the particular frame type that needs to be deployed for communication to take place. A homogeneous frame type is a necessity for two nodes to directly communicate even if their lower layer topology and higher layer protocols are the same. This is always a requirement to facilitate node-to-node interoperability across networks. Without homogeneous frame type, an intermediate device, such as a router or translation bridge, must be used to modify the packets when they transverse between communicating nodes.

Token-Ring frame types defined by Novell are Token-Ring and TOKEN-RING_SNAP. TOKEN-RING uses the IEEE 802.2 header at the data link layer. SNAP stands for SubNetwork Attachment Point and is 802.2 with additional bytes allocated for SNAP information. IBM PC Network follows the IBM Token-Ring pattern with an offering of IBM-PCN2_802.2 and IBM-PCN2-SNAP frames [- underscore in -802.2 and -SNAP].

Ethernet frame types include: ETHERNET_802.2, ETHERNET_802.3, ETHERNET_II, AND ETHERNET_SNAP. ETHERNET_802.3, sometimes referred to as "ETHERNET RAW" (in IEEE parlance, meaning there is not a standard header at the data link layer), is the traditional NetWare frame which predates acceptance of the IEEE 802.2 standard. Novell has changed the Ethernet default for SPX/IPX to 802.2 although 802.3 (RAW) is still widely used on enterprise networks. FDDI follows a pattern similar to a cross between Token_Ring and Ethernet with FDDI (RAW), FDDI_802.2, and FDDI_SNAP frames.

PROTOCOL ties protocols to a particular frame. The hexadecimal number in the sample PROTOCOL line is the protocol ID for the specified frame type. There can be multiple protocols per given frame type just as there can be multiple logical adapters (frame types) per physical board. In addition, a particular protocol may be bound to more than one frame type. SPX/IPX supports all of the frame types mentioned above. It is not unusual for a protocol to be capable of supporting more than one frame type per topology. However, the practice is to configure SPX/IPX with TOKEN-RING frame on Token-Ring, with ETHERNET_802.3 or ETHERNET_802.2 frame type on Ethernet, and with FDDI_802.2 on standard FDDI. TCP/IP runs the TOKEN-RING_SNAP frame type on Token-Ring, ETHERNET_II frames on Ethernet, and FDDI_SNAP on FDDI. Refer to Table 1 for a list of topologies, frames, and transport protocols.

Communication buffers may be tuned for efficiency via the LINK SUPPORT option. For best performance, the idea is to match the size of the largest possible packet to be sent or received, and to have enough buffers to keep up with the traffic.

```
LINK SUPPORT
    BUFFERS 40,1514
```

For Ethernet, we have defined 40 buffers at the optimum 1514 byte size. An optimum configuration on Token-Ring is:

```
LINK SUPPORT
    BUFFERS 14,4210
```

An upper limit of 64K bytes applies to number of buffers times buffer size.

Topology	Frame Type	Protocol Stacks
Token-Ring	Token-Ring (802.2)	SPX/IPX, NetBEUI, LLC
	Token-Ring_SNAP	AppleTalk
Ethernet	Ethernet 802.3 (RAW)	SPX/IPX
	Ethernet 802.2	SPX/IPX, NetBEUI
	Ethernet II	TCP/IP
	Ethernet_SNAP	AppleTalk
FDDI	FDDI (RAW)	-
	FDDI 802.2	SPX/IPX, LLC, NetBEUI
	FDDI_SNAP	TCP/IP
	AppleTalk	

The most popular topologies, frame types, and protocol suites listed above. The protocols per frame type are shown as they typically run on end-user networks. In reality, protocols such as SPX/IPX can be bound to many different frame types. Notice that Novell's SPX/IPX and IBM's protocols (NetBEUI and LLC) utilize the IEEE 802.2 format. Asynchronous Transfer Mode (ATM) is expected to support 802.2 and SNAP for multiprotocol virtual circuits.

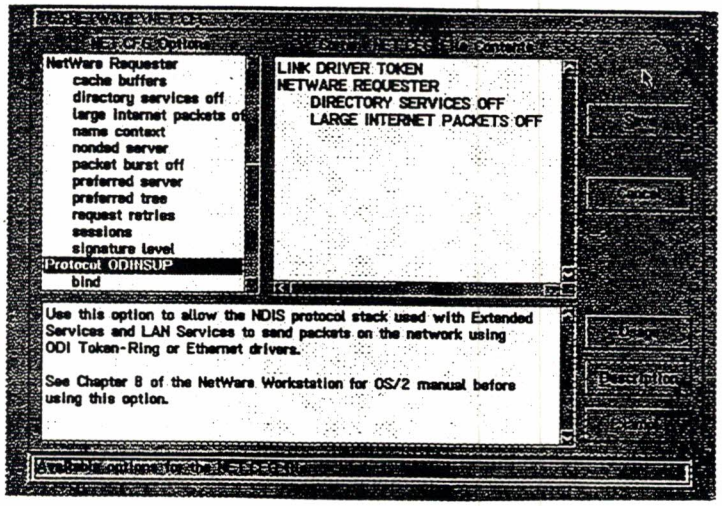
Table 1:DRIVERS, FRAMES, AND PROTOCOLS

PROTOCOL PARMS

The *config.sys* driver, *route.sys*, works with Token-Ring drivers and facilitates source routing across Token-Ring Medium Access Control (MAC) layer bridges. *Route.sys* enables source routing information to be placed into the packets transmitted from the NetWare Client running on the OS/2 node.

```
PROTOCOL
    ROUTE SOURCE ROUTE def gbr mbr
    NODES n BOARD n
```

def enables all routes broadcasts rather than single routes (which is the default). By sending all routes, broadcasts increase network traffic on any network containing redundant paths. All routes broadcasts are particularly detrimental on large networks. gbr broadcasts to all destinations across all rings via all routes. mbr makes the node send multicast broadcast frames as all routes broadcasts also. Some network gurus believe that Novell should not have placed these potentially detrimental options in the hands of the users at their workstations.



NODES n controls the number of entries in the OS/2 node's source routing table, where n is a numeric value between 8 and 255 (the default equals 16). **BOARD n** represents the logical board to which the **SOURCE ROUTE** settings apply. *route.sys* must be loaded for each frame type initialized on every physical Token-Ring network board installed in the workstation.

NDIS support on top of ODI drivers is initialized via the **PROTOCOL ODINSUP** option. In the example shown below, the Token-Ring driver, *token.sys*, provides the interface to the physical adapter and thereby controls transmissions on the network from the OS/2 node. Here one adapter driver binds to ODINSUP; however, the upper limit is four ODI drivers.

```
PROTOCOL ODINSUP
    BIND Token 1
```

IPX is configured by the *net.cfg* section shown below. The protocol may be bound to a specific network adapter driver through **BIND**. Bytes of memory are then defined for the router memory pool which handles network routing requests for ongoing communication connections. 150 bytes should be sufficient to handle several network boards in the workstation. Three sockets are needed per connection with each file server. Additional sockets will be needed for peer-to-peer named pipes communication. Number of sockets should not be less than 32; 128 is the maximum value.

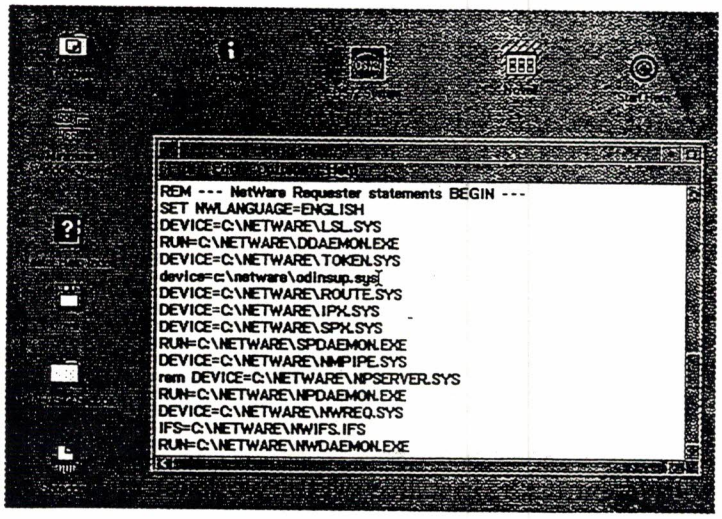
```
PROTOCOL STACK IPX
BIND token
ROUTER MEM 150
SOCKETS 128
```

SPX settings are a bit more extensive and primarily regulate timing. For the most part, default values were used in the example which follows. The **VERIFY, LISTEN,** and **ABORT** timeout values in milliseconds maintain or abort **IPX** connections and **SPX** sessions via "keep connection alive" packets. If the server does not receive responses to "keep connection alive" packets sent to an idle workstation, then the server will terminate the workstation's connection. Retry count sets the upper limit on resends to another computer which has not responded to a request. Send timeout specifies wait time between transmissions and will need to be increased if **SPX** connection errors occur over the network. SESSIONS sets the node's maximum number of concurrent **SPX** sessions.

```
PROTOCOL STACK SPX
VERIFY TIMEOUT 3000
LISTEN TIMEOUT 6000
ABORT TIMEOUT 30000 RETRY COUNT 15 SEND TIMEOUT 600 SESSIONS 400
```

Novell NetBIOS rides above IPX in the protocol stack. However, it cannot communicate with IBM NetBEUI which interfaces with LLC and which does not include IPX in its stack. As a result, many installations with non-standardized protocol environments run Novell NetBIOS simultaneously with IBM NetBEUI on the same node. By running both protocols, the node can communicate with all variations of NetBIOS across the enterprise. For example, a Lotus Notes communication server might need to interoperate with different workgroups. Some of the workgroups might be running Novell NetBIOS while others are running NetBEUI. If the communication server is configured with dual NetBIOS/NetBEUI, it will be capable of serving and linking both workgroups within the Notes application.

The ability to run dual NetBIOS/NetBEUI stacks is a new development under OS/2, v2.X and could not be facilitated under OS/2, v1.3. Refer to "Grand Pas de Deux" in the January 1992 issue of LAN magazine for details. While the dual NetBIOS/NetBEUI configuration is a technical milestone, its time has come and gone. The need to run either NetBIOS or NetBEUI is progressively diminishing as those protocols are replaced for all



practical purposes by native SPX/IPX or TCP/IP. Support of NetBIOS or NetBEUI on ever enlarging enterprise networks frequently becomes a network engineering nightmare due to routing problems.

Timeout settings control Novell NetBIOS connections in a manner similar to SPX. **BROADCAST COUNT** sets the number of times NetBIOS broadcasts a query for a particular name.

Internet extends the broadcasts beyond the local network and into "internet" territory. **BROADCAST DELAY** (in milliseconds) and **BROADCAST COUNT** default values double when internet is on in order to provide additional resiliency.

```
NETWARE NETBIOS
ABORT TIMEOUT 30000
LISTEN TIMEOUT 6000
VERIFY TIMEOUT 3000
INTERNET on
BROADCAST COUNT 4
BROADCAST DELAY 2000
COMMANDS 32
SESSIONS 16
NAMES 24
RETRY COUNT 20
RETRY DELAY 500
```

COMMANDS and **SESSIONS** specify how many **NetBIOS** commands and sessions can be supported concurrently by the NetBIOS driver. **NAMES** sets the maximum number of name table entries for remote workstations. **RETRY** parms establish the maximum number of retries before timeout and the delay between retries in milliseconds.

Novell named pipes rides the **SPX** connection-oriented protocol. It is an important and still widely used transport protocol in client-server computing involving OS/2 database server backends. However, like NetBIOS, named pipes is slowly being replaced by native **SPX/IPX** or TCP/IP.

CLIENT SESSIONS is the maximum number of client-to-server named pipes connections. On the other hand, maximum server-to-client connections depends on **SERVER SESSIONS.** The SERVER SESSIONS may be as large as 1000 but the practical upper limit advised by Novell is 400. Since Named Pipes utilizes **SPX** in the layer below it, **SPX** sessions must be greater than or equal to Named Pipes **SERVER SESSIONS.** Otherwise, Named Pipes **SERVER SESSIONS** will be limited to the number of **SPX SESSIONS. SERVICE THREADS** defines the number of threads which a Named Pipes server can use to handle requests from all clients.

```
NAMED PIPES
CLIENT SESSIONS 16
SERVER SESSIONS 400
SERVICE THREADS 32
```

The illustrated **NAMED PIPES** lines are designed for a named pipes server running as an application backend.

REQUESTER READJUSTMENTS

Requester parms are particularly relevant to performance and security between the NetWare Client and file server. **CACHE BUFFERS** are used to cache data from open files across the network. They help reduce network traffic by holding data in the OS/2 workstation's memory and thereby improve performance. The cost is workstation memory. Buffer size is automatically set to the maximum value that was negotiated between workstation and server at establishment of the communication connection. In the example shown below, we increased the number above the default of 8 buffers. A value of zero buffers specifies no caching.

DIRECTORY SERVICES OFF should be used if NetWare Directory Services (NDS) is not deployed. Otherwise, the workstation will waste time and packets looking for a domain controller. LARGE INTERNET PACKETS, known as LIP, is a very important consideration in routed networks. NetWare was originally designed to default to 576 byte maximum data packets (containing 512 bytes of data) for through router traffic. The new default in favor of large packets causes the presence of a router to be ignored during the negotiation of maximum packet size between workstation and server. Large packets mean faster performance of tasks requiring multiple reads or writes of files, and less packet traffic to trigger network bottlenecks. Large packets are enabled unless the line **LARGE INTERNET PACKETS OFF** is included in the **NETWARE REQUESTER** section of the *net.cfg* file.

```
NETWARE REQUESTER
CACHE BUFFERS 15
DIRECTORY SERVICES on
LARGE INTERNET PACKETS on
NAME CONTEXT "Katharine Panza.OS2.Miller Freeman SF"
NONDED SERVER (not applicable)
PACKET BURST on
PREFERRED SERVER Miller-Freeman-SF1 [-  underscore]
PREFERRED TREE OS2
REQUEST RETRIES 10
SESSIONS 8
SIGNATURE LEVEL 3
```

NAME CONTEXT makes it easier for Katharine to access her work areas in the Miller Freeman domain as defined by **NDS. NONDED SERVER** would not be used unless our node was to be both a NetWare workstation and a NetWare OS/2 file server. **PACKET BURST,** like LIP, is on by default. It is the most important parameter for increasing overall throughput across the network. Multiple IPX NetWare Core Protocol (NCP) reads and writes may be sent to and from the workstation without the "governor" limitation of a ping pong effect. The ping pong effect occurs when only one packet at a time is sent in each direction causing greater wait time and

more packet traffic. In simplistic terms, request-reply, request-reply, request-reply, etc. communication between two nodes is replaced by a request, reply, reply, reply, etc. exchange. Packet burst is triggered upon initiation of a data operation three times the maximum packet size negotiated between the workstation and server.

PREFERRED SERVER designates a server for initial attachment by the workstation. **PREFERRED TREE** enables the workstation to connect to the *OS2 magazine* tree, assuming that there is a separate tree for each magazine at the Miller Freeman location in San Francisco. The maximum number of retries before timeout is controlled by **REQUEST RETRIES. SESSIONS** sets the maximum number of connections which the workstation can have with all file servers.

Signature level may be used to implement a high level of security on the network via packet signature. Packet signature is designed to prevent the creation of fraudulent packet traffic to access NetWare file servers. A number ranging from 0 to 3 regulates the packet signature feature at the workstation as follows. With 0, the workstation does not sign packets; 1 (the default) causes the workstation to sign packets only if the server requests a signature; the setting of 2 has the workstation always signing packets if the server is capable of signing; 3, the highest security level, requires the workstation to sign packets transmitted by the workstation, and also requires the server to sign packets it transmits. Under 3, communication will fail unless all packets in the two way conversation are signed.

MM 1210
Producing Your Own CD-ROMS

Guy Wright
Hourglass
San Francisco, Calif.

Introduction

This class is for anyone interested in learning about producing their own CD-ROMs.

The Four Elements

- CD-ROM Applications, Advantages and Disadvantages
- CD-ROM Technology
- CD-ROM Production Process
- CD-ROM Finances

CD-ROM Applications

- Archiving data
- In-house Software/Data distribution
- Commercial Uses
  - Commercial Software
  - Demos
  - Kiosks or Data used in other applications (fonts, clip-art, databases)

Advantages of CD-ROM

- Capacity
  - 680 Mb data, 74 Minutes of audio or MPEG video
- Durability
  - Estimated life of up to 100 years
- Hardware platform independent
  - any CD-ROM drive can read any ISO standard CD-ROM disc (although the computer doing the reading may not be able to understand the data format)
- CD-Audio, video and multimedia aspects
- Inexpensive
- Rapidly becoming the standard delivery format for large applications

Disadvantages of CD-ROM

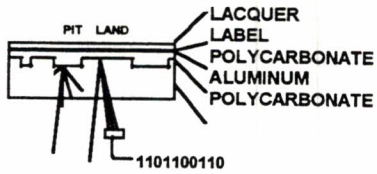
- Slow data access
- Not appropriate for all applications
- Not everyone has a CD-ROM drive
- Somewhat restrictive guidelines -
  - 8.3 file names,
  - limited to upper-case letters, numbers, period, and underscore,
  - 8 levels of sub-directories
- Still something of a mystery

A Brief history of CD-ROM Technologies

- CD-Audio Red Book introduced 1980
  - Sony and Philips
  - 1982 CD-Audio Discs begin shipping
- CD-ROM Yellow Book introduced 1985
  - High Sierra group meets
  - ISO (International Standards Organization) adopts recommendations which become ISO 9660

How a CD-ROM Works

Laser light is shined through underside of disc as it spins. When the beam hits a 'land' it reflects enough light to trigger the photo detector. When the beam hits a 'pit' the light is dispersed and does not reflect enough to trigger the detector.



Triggers are translated into a series of binary digits, then sent to D to A converter for audio or straight to a computer as data.

CD-I (Green Book) and other Set-top Boxes

- 1983 Philips started work on CD-I
- 1990s CD-I, CDTV, FM-Townes, 3DO, Jaguar, Sega-CD begin shipping
- Custom or proprietary operating systems, expensive to develop for, limited markets
- Whole new set of ISO modifications (CD-I Green Book, CD-XA, CD+G, MPC,...)

Photo CD (Pink Book)

- Kodak introduces Photo-CD technology
- First popular multi-session specification

MPEG (White Book)

- 74 minutes of video
- MPEG Video compression scheme
- Usually requires special hardware decoder

CD-Recordable (Orange Book)

- 1989 Meridian systems cost $100,000 + (gold discs cost $100 each)
- 1991 Sony brings price down to $28,000
- 1993 Frankfurt Group proposal accepted by ECMA (European Computer Manufacturers Association)
- 1993 Sony, Kodak, Yamaha, Richo, JVC, and others are shipping CD-Rs
- CD-R's now cost from $2,000 - $9,000 (gold discs cost under $10)

Multiplatform, Multisession, Mixed-mode, Bridge Discs

- Multiplatform - Includes versions for multiple operating systems on single disc
- Multisession - Allows recording to the same gold disc incrementally
- Mixed-Mode - Both CD-audio and CD-ROM data on same disc
- Bridge-Discs - Combine features of different formats

CD-ROM Production Process

- Determine Appropriateness
- Determine kind of uses
- Compile all data
- Build (or purchase) Front-end
- Build (or purchase) Search Engine
- Programming, Linking, Testing
- Pre-master
- Test
- Duplication
- Distribution

Determine Appropriateness of CD-ROM

- While CD-ROM is great for a lot of applications it isn't right for everything

Appropriate CD-ROM Applications

- Archival
  - durable media not affected by magnetic fields, temperature, moisture, power fluctuations, etc.
  - less bulky than other archival media
  - read only means no accidental erasure
  - estimated life of up to 100 years
- Distribution
  - 650 MB on a single disc
  - play back on any PC with a CD-ROM drive
  - lower shipping costs
- Commercial
  - lower duplication costs for large applications
  - growing acceptance of CD-ROM
  - 'high-tech' perception

Inappropriate CD-ROM Applications

- Frequently Updated Information
- Fast access required
- Read-Write required
- Small applications

Determine Kind of Application

- Who will use the product?
  - Will they need special instructions or generic instructions?
  - End user, professional?
  - Will they already know how to use the disc?
- How will they use the product?
  - Is this a temporary product?
  - Will it be used daily, weekly, yearly?
- Where will it be used
  - in-house only?
- How frequently will the data need to be updated?

Compile all data

- Gather all materials
- Re-keying (if necessary)
- Scan or generate graphics (if necessary)
- Format conversion (if necessary)
- Investigate legal issues

Build (or purchase) Front-end

- The user will have to interact with the data somehow
  - No front-end
  - Limited front-end
  - off-the-shelf front-end
  - custom front-end

Build (or purchase) Search Engine

- The user will have to get to the data somehow
  - simple directory structure
  - use run-time viewer
  - license a commercial search engine
  - develop your own search engine

Programming, Linking, Testing

- This is where it all comes together on hard disk or network
- Search index generated
- Final data linked to front-end
- Test, test, test
- Test some more

OS/2 Specific Concerns

- ISO level 1 does not support HPFS
- MSCDEX only supports ISO level 1
- WARP may not accept ISO level 3 DLLs
- Limited number of OS/2 specific Premastering packages

Pre-master

- Should reveal illegal file names
- Uncovers ISO problems
- Emulator may uncover timing problems
- Create ISO image for dumping to tape or hard disk

Gotchas

- zero out hard disk
- don't leave sensitive data on the disc
- delete fonts, clip-art, or other applications
- don't reference files with illegal names
- don't hardwire code to specific drive
- don't try to write to the disc

Gold disc

- send raw data to service bureau
- send formatted data to service bureau
- burn gold disc on CD-R

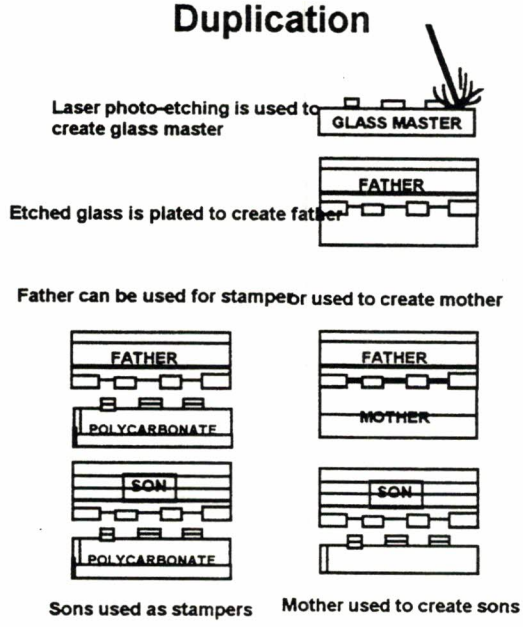
Testing

- Test gold disc on target systems
- If everything works
  - if you only need one copy then you are done
  - if you only need a few copies then you can burn more gold discs
  - if you need more than a few copies go to duplication
- If there are problems
  - go back to pre-master stage
  - go back to programming stage

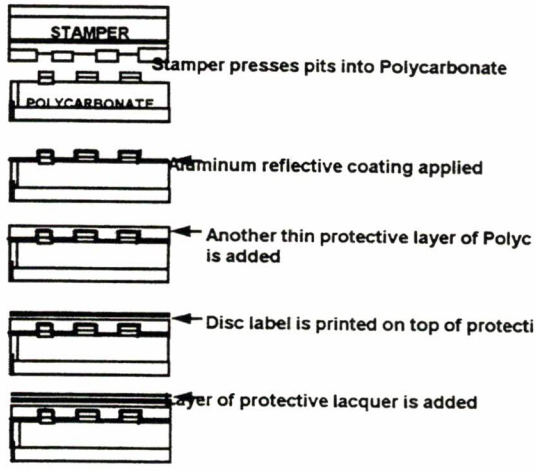
Duplication

- Pre-master (if it hasn't already been done)
- Glass master
- Father
- Mother
- Sons
- Discs
- Insertion of discs, tray cards, inserts into packaging
- Shrink wrapping

Duplication



Duplication



CD-ROM Finances

- preparation
- software
- hardware

Data Preparation

- Compilation of data
- Custom graphics, animation, sound, video, etc.
- royalties

Software

- programming
- search engine
- run-time licences

Hardware

- Duplication
- Printing
- Shipping

Duplication

- $500 - 1500 Mastering (depending on quantity)
- $.75 - 1.50 per disc (depending on quantity)
- $.05 per disc for hand insertion in custom packaging
- $.05 per disc shrink wrapping
- $.05 per disc per extra color on label

Packaging

- $.30 standard jewel case
- $.25 - $1.50 for custom packaging
  - orders usually 10,000 or more

Printing

- Label
  - artwork
  - film (film positives)
- Tray card and insert
  - artwork
  - film
  - printing $400 per 1,000

Shipping

- Jewel Case
  - $.05 @ for cardboard mailer
  - approx $1.50 @ postage
- Custom ROMvelope
  - $.05 @ for hand insertion charge
  - approx $.50 @ postage
- Shipping of film and gold disc to duplicator
- Shipping of finished discs to destination

Conclusion

- CD-ROM is an excellent way to distribute, or archive data
- CD-ROMs are inexpensive
- It is not that difficult to produce your own CD-ROMs

MNC 1206

Intergrating NetWare Into the OS/2 Environment

No paper available. Please contact speaker

this page intentionally left blank

MNC 1207
Performance Tuning OS/2

Ron Cadima
IBM

ABOUT THIS DOCUMENT:

This paper makes the assumption that the reader is familiar with OS/2 Warp and many of the technical terms in use by the computer industry. It presents general and specific tuning information for a computer that will have or has had OS/2 Warp installed. It starts by taking the reader through some general system setup considerations with regard to processor type, disk type and the amount of memory in your system. Next it goes into detail with respect to CONFIG.SYS statements that should never be modified and those that can be modified to help improve system performance. System Settings and Desktop Settings are covered next. Finally Appendix A presents measurement information about how much memory is used for different functions and products.

Some terms and phrases that will be used in this document are as follows:

Warp - OS/2 Warp Version 3.0 both the version that is used with Windows already installed on your machine and the version that includes the WIN-OS/2 support.
OS/2 - Refers to the OS/2 Warp Version 3.0 product and its capabilities.
System - a generic term that refers to computer with an operating system and application programs installed on it.
Memory - The amount of physical memory or RAM that is installed in the computer. or used by a particular function.
FAT - File Access Table file system.
HPFS - High Performance File System
Desktop- Sometimes referred to as the Workplace, this means the Workplace Shell feature of OS/2. The Workplace Shell is the part of OS/2 which controls how you OS/2 systems appears on your computers video screen.

Trademarks:
IBM Corporation:
OS/2 Warp Version 3.0
DB/2
LAN Server 4.0
Personal Communications/3270
BonusPak

Microsoft Corporation:
Windows 3.1 Excel
Windows 3.11 Word For Windows
Windows for Workgroups 3.11

GENERAL SETUP CONSIDERATIONS:

There are some basic considerations that need to be made about the computer that you will install Warp on. It breaks down into 3 elements: processor , memory and disk.

In most cases the processor that you have in your system has a minimal impact on performance compared to the amount of memory and the speed of the disk. The only real considerations to make with respect to processor are its age. You want to avoid 386 and 386SX processors. You want processors that can be upgraded and those that support instruction caching sometimes referred to as Level 1 or Level 2 cache. There really is no optimum level instruction cache size. More instruction cache is usually better than less. The only real determinant should be costs.

Memory, also called RAM, is a different story. Without enough memory your system will run slow. If you cannot afford to buy memory for your system, then you must manage the things you install. You only want to install fonts, device drivers, objects and applications that you will actually use. You may think that just because you have not started an installed application, it is not using any memory. This is not always the case. Many newer OS/2 applications will register classes and objects with the Workplace Shell or add items into path statements, set statements and the like. Many of these things will cause memory to be used even when the program is not running. In some cases, the program will require special device drivers be installed which also use up memory.

Other things that affect memory usage are disk caches, CD-ROM caches, multimedia support and buffers. The important factor to consider for the disk cache, is that it is designed for general use. Since most systems are not general use, the disk cache should be tuned to the specific system environment to optimize memory usage. We will discuss the disk cache considerations in more detail when we look at the file system options.

Probably the most important piece of hardware on your system is the hard disk. It will affect the performance of starting your system, loading applications, the speed of applications, and the general performance of your system.

It is best to have a disk subsystem which uses a bus-mastering type of adapter. Many SCSI devices and some of the PCI devices have this capability. These types of devices allow for multiple requests to be sent to the disk device to be processed rather than just one command or function at a time. They also allow what is called scatter / gather capabilities. Without bus mastering, data that is transferred between the computer's memory and the disk must be in contiguous memory, one byte after the other. With bus mastering, the data does not have to be contiguous and does not have to be on a 64K byte boundary. Bus-mastering relieves the system of a lot of overhead and therefore performs faster in actual usage , although you will see little or no difference when running benchmarks..

When the memory in your system is over-committed, that is, the operating system and the applications you are running need more memory than is physically available in your computer, Warp will page, or swap, code and data that has not been accessed for a while to the disk in order to make room for the needed code and data. In this instance, the disk is the single most important factor in your system with respect to performance.

Other considerations with regard to the disk is whether or not it has its own caching. If it does and your applications perform mainly sequential disk access, then you can reduce the size of your software disk cache. Hardware disk cache will have little or no effect when the programs perform random disk operations. The hardware cache will improve the performance of starting the Warp operating system and the applications.

FILE SYSTEMS:

OS/2 supports two file systems for use on your hard disks: FAT and HPFS. No matter which file system is used, there are some basic considerations. You need to install the file system depending on which operating systems will access the data. If you plan to boot a DOS or Windows system natively, then any data that will be accessed must exist on a FAT disk partition. If you will only be running DOS and Windows applications in a Warp VDM (Virtual DOS Machine), then the file system can be HPFS or FAT. Also, when accessing a file on a server, and the server file system is HPFS, you do not need to install HPFS on your local client machine. HPFS only needs to be installed on a computer when a partition on a local hard disk is formatted as HPFS.

The amount of memory you have in your machine should affect the decision about which file system to use. When HPFS is installed, it requires a minimum 200 to 250k of working set, plus the space allocated for its cache. This is a large amount of memory and is the main reason why FAT is used when Warp is installed on a system that contains 4 MB of physical memory..

Regardless of which file system you select, you need to plan for future requirements: How will maintenance and

fixes be applied? Where will applications be installed? Where will data reside? etc. I recommend that when you set up your hard disk, you create a minimum of 3 partitions. One will be for the operating system(s), one for your applications and static data files, and another for dynamic data files and temporary files. Decide whether you want to use Boot Manager or Dual Boot. If you select Dual Boot, then OS/2 must be installed with the FAT file system.

For temporary files, it is better to have them pre-allocated and reuse them rather than create and destroy them every time. Extending a file can be almost twice as slow as just writing to a file. Also, if you reuse files, it will definitely reduce the fragmentation in your disk directories.

Now let's look at some specifics with regards to the two file systems.

FAT:

FAT is best suited for disk partitions that are 80 MB or less in size or that have a limited number of files installed. Usually, 256 files is a good target, with up to 500 acceptable. The number of files become important because FAT files are allocated based on a sector size. The sector size is determined by the size of the disk partition and can be 2K, 4K, 8K or higher. Since most file sizes are not an exact multiple of the sector size, disk space gets wasted. For example, installing DOS, Windows and Warp on a 100MB partition resulted in 2.2 MB of disk space being wasted.

Also, when Warp is allocating space for a file in a FAT partition, it will look for the largest available free space area to write the data or create the file. If lots of files were created and deleted, or expanded, these free space areas become smaller and smaller and are spread out over the disk. This is referred to as fragmentation. When a file is stored in many areas on the disk, it takes longer to read that data simply because the disk head has to do more seeking.

The FAT file system disk cache is defined by the DISKCACHE= statement in the CONFIG.SYS file. New in Warp is the "D" designation for the size of the disk cache. Warp will allocate space for the disk cache based on the amount of physical memory that is installed in the system. If more than 8 MB is installed, 10% of the physical memory will be used for the disk cache up to a maximum of 14.4 MB. Based on what applications and support you install in your system, this may be too high and cause you to over commit your memory. If there is not a lot of disk work done on your system, or you are using DB/2 or HPFS, then you should reduce the size of your disk cache. A number between 128 and 256K is sufficient for most systems. On a 4 MB system, we set the disk cache to 48K.

In conjunction with the cache size, you should also set the cache threshold. This determines which records being written or read will be placed in the cache. The default is 1 sector. If your cache size is 128K or larger, increase this value to at least 16 and preferably 32. More detail about the DISKCACHE= statement will be given in the section on CONFIG.SYS.

Some additional items for improving performance using FAT are:

* Group files by usage, the most used files first, and least used files last.
* Group files based on whether they are static or dynamic. All dynamic and temporary files should be placed in a separate partition.
* If possible, place all temporary files in a single directory.
* If possible, permanently create temporary files and reuse them.
* Only create directories that are really needed. The fewer number of directories you have to search, the faster your search will be.
* Defragment your partitions after installing new applications, deleting files or applying maintenance and fixes. Any DOS defragmentation program that is aware of hidden files can be used.

HPFS:

HPFS does away with some of the concerns that are prevalent with FAT. Files are allocated based on a 512byte granularity instead of a sector size, therefore fragmentation is greatly reduced. Also HPFS is especially efficient when handling large partition sizes, > 100 MB, and large numbers of files, > 500. One thing you should look out for is to not allocate more than 5000 files in a sub-directory or directory. When you exceed 5000 files, you will start to degrade performance. The HPFS file system shipped with the Warp product has a cache limit of 2 MB. There is no such limit when using the HPFS386 file system that comes with the LAN Server products.

HPFS does not have a separate line for the cache parameters. These are specified on the IFS statement in the CONFIG.SYS file. As with FAT, you should specify the caching threshold value. Use the /CRECL parameter to do this. Here you specify a number in terms of K bytes. 32K is usually a good starting value.

As mentioned earlier, the big drawback to HPFS is the amount of extra memory that it requires. Even if there is no HPFS partition on your system, it will cost between 200 and 250K in working set memory, as well as the space for the HPFS cache. If you are installing Warp on an existing DOS and or Windows system, you do not want to install HPFS. When your system is up and running, you can check the working set of your system. If there is enough free memory and you wish to create a HPFS partition, then you can use selective reinstall to install the HPFS support. Remember that any data stored in the HPFS partition can not be accessed if you boot your machine under DOS.

Another advantage of HPFS over FAT is in the area of extended attributes (EAs). EAs are data attached to a file and used to provide information about the file it is attached to. The name of an object that appears in an OS/2 folder or on the OS/2 Desktop is stored in EAs. In HPFS, EAs are part of the HPFS file control block which is read when the file is open. In FAT, EAs are stored in a separate file and require additional I/O to access them, and are therefore slower.

SYSTEM TUNING:

Before covering OS/2 system tuning let's review the concept of threads and how they are used in OS/2. Every program that you run on an OS/2 system will process as one or more threads. A thread is the basic unit of work within OS/2. Each thread requires a minimum of 4K stack space. Stack space is used to save parameters and addresses when different functions are called from a thread. These functions could be to reading from or writing to a disk, using semaphores, and other system and program functions. It is important to ensure that you allocate enough stack space for the thread to run. You can use programs such as EXEHDR.EXE to see and modify the stack space allocated for your program.

All programs, DOS, Windows and OS/2 use a minimum of one thread. Each thread executes at a given priority. The priority is used by OS/2 to determine which thread gets to run when more than one thread is ready to run. The highest-priority thread ready to run will be the one dispatched by the system and given time to run. It is given what is referred to a "time slice". This is a set period of time for which a thread is allowed to run. After this time has expired, the highest-priority thread that is ready to run will be given a time slice in which to run. If the same thread still wanted to run, and it was the highest-priority thread, it would receive another time slice to run in.

OS/2 programs usually run at normal priority, which is 200. Under program control, the program can change its priority to be server class, 300, or time-critical, 800. It can also change its subclass priority between 0 and 31. A thread that has priority 200 and a subclass of 15 has a higher effective priority than a thread with priority 200 and subclass 0, and therefore runs first. All DOS and Windows applications always run at priority 200. In OS/2 Warp, we have added the capability to define the subclass priority for these applications. Care should be taken when using this subclass priority since it may cause other programs to run slower or generate errors in the case of communication applications that do not get the subclass boost.

Warp will give OS/2 threads a priority boost for specific types of functions and states. Following are the priority boosts that are received, in descending order:

DISK I/O:

When an interrupt is received stating that a disk operation has completed, the thread that processes this state will receive a priority boost for one time slice to process this interrupt.

STARVED:

In the CONFIG.SYS file, there is a statement MAXWAIT=3. The number 3 specifies how long a thread can be in the ready-to-run state without having received time to run. If a thread had been waiting to run for 3 seconds, its priority would be raised to the Starved priority level.

KEYBOARD:

This is a boost in priority given to a thread when it is interacting with the system keyboard. Basically, it is for accepting typed in data.

FOREGROUND:

This is a priority boost given to all threads of the program that owns the active window on the OS/2 screen, the one that is highlighted.

WINDOWED:

Threads which have windows showing on the display will also receive a priority boost when drawing or writing into that window.

When a thread is in more than one of these states, the system will combine the states and give a priority based on the combined states.

For DOS and Windows applications, these priority states do not apply. Instead, a second thread is used to handle the interrupt, and that thread is given an interrupt boost..

Idle class is the other priority class that exists in OS/2. This priority level runs only when nothing else in the system wants to run. Threads in this class will not receive any of the state boosts. If DOS and/or Windows applications are being run, then you should avoid the use of threads that run in idle class because they will never run but will take up system space.

Now let's look at specific things in the CONFIG.SYS file that you can change to affect the performance of your system. First we cover those statements that you should NOT modify unless you have a system which has a special use, such as a process control system. An example would be a system used to monitor a manufacturing machine or chemical process, where timing and response time are critical factors.

PRIORITY_DISK_IO=YES

This allows the application that has screen focus to receive a priority boost when it's disk operation is complete. This applies to the first time slice given to the thread after the disk operation is complete. After the time slice, the state is reset for the thread and the priority boost removed.

MEMMAN=SWAP,PROTECT

Allows OS/2 to swap code and data to disk when more memory is needed than is physically available in your computer. If you do not specify SWAP, you will need enough physical memory in you computer to hold all the code and data that OS/2 and your application needs to run. You do not pay a penalty by specifying SWAP and then not needing to use it.

The PROTECT parameter allows Dynamic Link Libraries (DLL) to allocate protected memory. Protected memory is memory that is protected from being accessed by unauthorized programs.

You can also specify a COMMIT parameter. This forces the system to ensure that enough physical memory or swap file disk space is available for a memory object when it is created. Normally, OS/2 commits physical and disk memory when the page of memory is actually touched. Specifying this parameter can significantly increase the amount space your swap file uses on disk.

TIMESLICE=X,

This statement is not found in the CONFIG.SYS file after you install OS/2 but is sometimes recommended to be added. This was okay to add for OS/2 2.0, 2.1 or 2.11 systems, but not for OS/2 Warp. In Warp, we dynamically modify a threads time slice based on actions that have occurred. For instance, if a thread took a page fault during it's time slice, we give it an extra time slice to process what is contained in the faulted page. We also give applications doing disk I/O extra time slices if the data they are reading is in the disk cache. When the TIMESLICE= parameter is used, none of these actions will occur. Instead, each thread will be given the minimum time slice of X, and its time slice will not be allowed to go beyond value Y.

PRIORITY=DYNAMICIf DYNAMIC is not specified, then each thread will receive a minimum and maximum time slice of 32 milliseconds with no additional time slices as described in the TIMESLICE= statement description above.

DEVICE=C:\OS2\BOOT\VDISK.SYS

This statement allocates a virtual disk in your computers physical memory. It is used for quick access to often-used files and programs. This was good for a DOS environment where the extra memory in your computer was not used by DOS, but OS/2 uses this memory, and your performance can be adversely affected if a VDISK is used. It is much better to increase the size of the disk cache if you have unused physical memory than it is to use VDISK.

Now let us look at statements in CONFIG.SYS which you should change to help improve the performance of your Warp system.

LIBPATH=

This tells the system where to find DLL files and disk device drivers that Warp and applications use. Place the directory names in order of usage. The most accessed directory should be first, the least used last. If possible, place the DLL used by a program in the same directory as the working directory when the program is running. Then, you do not need to add that directory to the LIBPATH statement. Also, place all directories that are on a network at the end of your LIBPATH statement in case the network goes down and they cannot be accessed. See the discussion of dynamic LIBPATH support later on in this paper for additional considerations when accessing network drives.

PATH=PATH is used to specify where Warp searches for program executable files, EXE, COM, CMD, BAT, etc. Place the directories in order of most used first, least used last. If programs will be executed from an object on your desktop or folder, specify the path there and not in the PATH statement. Only place directories in the PATH statement for executable that will be called from other programs or commands.

DPATH=

The same principle applies to DPATH as PATH and LIBPATH. For resources that applications use, place the most used directories first and the least used last.

BUFFERS=90

Buffers are physical memory used to support partial sector reads and writes in a FAT file system environment. They are also used to cache FAT directory entries and for swap file disk I/O. Because BUFFERS are used to cache FAT directory entries, this number should not be reduced below 60, unless you are not using the FAT file system on your disks. Reducing this number will increase the number of disk reads that are done to the FAT directory entries and therefore slow down your system.

MAXWAIT=3

This specifies the maximum amount of time that a thread will be in a ready-to-run state without receiving a time slice to run in. After this time expires, the thread will be given a boost in priority so it gets a chance to run. Reducing this value to 2 may help in systems where there are a lot of programs running, or multiple separate DOS/Windows programs running. Reducing it to 1 on systems where there is swap activity taking place can slow down the system.

DISKCACHE=D,LW,T,AC:

This is used to specify the amount of physical memory set aside to cache data that is being read from or written to disk. D represents a dynamic disk cache value that varies in size from 48K to 14400K, based on the amount of physical memory you have in your system. If you have more than 8 MB in your system, D will be 10% of your system's physical memory. The D should be changed to the actual amount of disk cache space you require. If you have a system where your programs do not do much disk I/O or where your memory is being used up by your applications, you may want to set this value to 128. Also, reduce this value if you are using the DB/2 product or any other product that provides its own disk data caching.

The T parameter is not defined in the default CONFIG.SYS file. It defines the cache threshold for records that go into the cache. The default value is 4. This means that any record that is 4 sectors or less will go into the cache, while those greater will not. If your disk cache size is 128K or larger, add this parameter to the disk cache statement. If known, set this value to the largest record size used. Otherwise, set the value to 32.

The LW parameter activates the Lazy Write or write behind feature. This allows the application to get control back before the data is actually written to disk. A separate thread will write the data from the cache to the disk when necessary or opportune to do so. You should always use this option, and code your applications to open files with a cache bypass option if disk data security is imperative.

SWAPPATH=D,R,S

SWAPPATH specifies where code and data pages are swapped to on disk when more physical memory is needed than is available in your system. D represents the path where your SWAPPER.DAT file is located. For systems which have multiple partitions or multiple disks, this should be placed on the least used directory of the least used disk. Also, try to physically locate the swap file on the disk based on its usage. If you are doing a lot of swap activity, place the swap file at the start of the disk. If is it rarely used, place it at the end.

S is the size that the swap file is initialized to when you start your Warp system. Make this large enough so that it does not have to grow in size while you are running your programs. You should perform your normal computer functions and look at the size of the swap file when you have the most activity. Then set the value of the S to this size in the CONFIG.SYS file. If you are using the FAT file system, IPL your system under DOS, delete the SWAPPER.DAT file, defrag the disk partition where the swap file will be located, and then IPL your OS/2 system. This should keep your swap file from getting fragmented.

R specifies the amount of free space that must be in the swap file. The default for this value is adequate and only needs to be changed if you want to be warned earlier about a possible out-of-memory situation.

THREADS=

THREADS defines how many threads the system will be able to use. One page of resident memory is need for approximately every 32 threads that are defined. As a minimum, you will need 80 threads to support the base Warp system and 3 or 4 OS/2, DOS and or Windows applications. The system will support up to 64000 threads, but typically you will not have enough memory in your system to support more than 300 to 500 threads. 18 threads are required for LAN Server 4.0, and 12 for Personal Communications/3270. You will need an additional 2 threads for each Personal Communications/3270 session that is started.

To calculate the number of threads that you will need in your system, use the formula 54+(2xN)+10 where N is the number of programs that you will run together. If the program requires more than 2 threads, add in the additional threads.

ADDITIONAL CONFIG.SYS CONSIDERATIONS

You should only install the devices drivers that your programs actually need to run. Do not install extra communication, printer, video or device drivers if they will not be used. Below are a list of device drivers and virtual device drivers that are normally found in CONFIG.SYS and that may not be needed. (Virtual device drivers are used to support DOS and Windows applications and usually have a V at the beginning of their name.)

VEMM.SYS Used to support Extended Memory use in DOS and Windows programs.

VXMS.SYS Supports Extended Memory in DOS and Windows applications.

VDPMI.SYS Supports the DPMI memory access and is required to support all Windows applications.

VW32S.SYS Supplies support for the WIN32S Windows APIs. Not needed for Windows programs that do not use WIN32S APIs.

IBMxFLPY.ADD x will be either a 1 or a 2. 1 is used for family 1 machines and 2 is for MCA machines. You do not need both.

XDFLOPPY.FLT This is required for reading diskettes which are written using the XDF (Extended Disk Format) format. These are usually OS/2 and PCDOS 7 installation diskettes, printer and video driver diskettes and possibly CSD diskettes. The Warp Install and Disk 1 diskettes do not use the XDF format.

COM.SYS and VCOM.SYS are only required if you will be doing serial or async communications.

If you use the selectable CONFIG.SYS option in the Archive and Retrieve feature of Warp, it is possible to create multiple CONFIG.SYS files which will contain different device drivers based on your needs. For example, you could create one CONFIG.SYS file that is used for maintenance and installation. This one would include the XDFLOPPY.FLT and both IBMxFLPY.ADD drivers plus additional threads. Another CONFIG.SYS used for normal processing could have these device drivers removed and a lower number of threads. When maintenance has to be applied to the system, or some additional new products installed, you could IPL the machine with the Archive and Retrieval menu active and select the proper CONFIG.SYS to be used. An extension of this would be to archive the CONFIG.SYS and the OS/2 INI files and select those to apply maintenance.

SYSTEM SETTINGS

Let's look at some of the System Settings and Desktop settings that can be tuned to improve a system's performance. In the System Setup Folder there are many utilities that are available.

SPOOLER:

The drop-down menu provides an option to disable the spooler. This can be done without problems when only one job will be active on your printer at a time. This will save a little bit of memory and one process and thread in your system. If you have a dedicated print spool machine, then you should raise the priority of the spooler to it's highest point. For non-dedicated print spool systems, the priority should remain at the default. You would only increase this if (for example) you raised the subpriority of a DOS session and wanted to print documents while running the DOS session. In this case, you would want to make the priority of the print spooler the same as the DOS session.

SCHEMES AND COLOR PALLET:

You should use solid colors and avoid the use of bitmaps for desktop and folder backgrounds. These particular options use more memory and require more processing time to display them.

SOUNDS:

Deselect the System Sounds options, unless you like the noises when opening and closing your folders. It costs between 250 and 300K in working set just to hear the noise. An additional 40K or so of working set can be saved by executing DINSTSND.CMD in an OS/2 command session. This will unhook the system sounds from the OS/2 desktop. To get them back, you would execute INSTSND.CMD.

FONT PALETTE:

Only install the fonts that you will actually use in your system. Also, try not to mix fonts in folders or on the OS/2 Desktop. If you do, you will be using extra memory. Outline fonts tend to be a little smaller in terms of memory than bitmap fonts and once they are in the cache, they perform just as fast.

WIN-OS/2:

When defining a Windows program, select common sessions and make DDE and Clipboard support both private. If floating-point is used in your Windows applications, use the enhanced run mode for Windows. Also, if you

have a Windows application that you always run, it is better to start it via the Startup Folder than to use the Fast Load option and start it from the Desktop. Finally, ensure that when you migrate, install or add Windows programs to you system, that the run mode for these is the same as the common session run mode. If run modes are different, then you will have two separate Windows sessions running even though you stated that a common session is to be used.

SYSTEM SETUP:

Disabling the Animation and Print Screen option can save a little in code path and memory. Setting the System Logo option to none can save some time when loading applications that check this parameter to see how long to display their applications logo. The type-ahead option will use a little more memory. When selecting screen resolutions, remember that the higher the resolution, the more memory that is used. Very high resolution and color support can require 100 to 200K of physical memory.

MOUSE

Mouse pointers are basically bitmaps. The amount of memory used will be affected by which mouse pointer style you choose. If you activate the comet cursor, this will cost additional memory and processing time whenever the mouse is being used.

DESKTOP SETTINGS:

Desktop settings are selected by choosing the Settings option on the Desktop menu. Choosing Automatic Lockup will cause extra code path and processing time to occur. Normally this is very minimal, until the time-out value expires. Then you are looking at about 40 to 50K of working set.

If you select Create Archive on system restarts, this will slow down your systems IPL time. You want to set this option only when you have changed your desktop and/or configuration and want to save it. Once it has been saved, turn this option off. You can get to the Archive Retrieve menu screen when you IPL your machine by using the ALT+F1 key sequence.

CONCLUSION

All the bells and whistles, neat features, applications and devices come at a cost. They all cost disk space, they all use memory, and they all have the potential for slowing your system down. To minimize this, plan ahead. Decide what the system will be used for and what your future uses will be. Then, set up the system accordingly. Install only the things that you are actually going to use and need. Ensure that there is a match between the software you install and the hardware you install it on. If an application or feature cost more memory than you have, either don't install the feature, or get more memory for your system. Finally be creative and logical. You can set the system up with different configurations and different support, defined to optimize performance based on the particular functions you perform and applications you use. Become aware of what Warp has to offer and use it to your advantage.

APPENDIX A:

The memory measurements results presented here are for planning purposes only. There is no guarantee, implied or otherwise, that you will reproduce these numbers exactly on any other system. Measured results will change based on the hardware configuration of your computer, and what software is installed in the system. If the hardware or software configuration changes, the results will also change.

All numbers are presented in megabytes of memory, and indicate all the physical memory that is used to execute

a given function. The numbers include both Warp system code and application code.

All measurements were made on a PS/2 A21, with 26MB of memory, VGA video, ESDI disk, mouse, keyboard and 16/4 Token Ring adapter. HPFS and Multimedia support were not installed in the base system. The CONFIG.SYS file was modified to be equivalent to how it would look on a 4 MB system. DISKCACHE=48, THREADS=96, and no IFS statement for HPFS.

Memory Requirements:

Function	OS/2 Warp For WIndows	OS/2 Warp With WIN-OS/2
Open System Folder	3.6 MB	3.7 MB
Open Command Prompts	3.5	3.6
Start a DOS Session	4.5	4.6
Exit the DOS Session	4.3	4.3
Start a Windows Session	6.3	6.0
Exit the Windows Session	5.3	5.3
Open Productivity Folder	3.5	3.6
Start Enhanced Editor	4.5	4.6

The numbers for DOS and Windows sessions are for full-screen sessions and therefore include the working set memory for switching back to the Warp Desktop to stop the measurement process. Following are numbers for starting a DOS session in a window and the Windows PROGMAN support in a window, also referred to as a seamless session.

Function	OS/2 Warp For WIndows	OS/2 Warp With WIN-OS/2
Open DOS Window Session	4.16 MB	4.15 MB
DIR Command in DOS Window	3.00	3.00
Windows Session with Standard Run Mode and Public DDE and Clipboard Support		
Start PROGMAN	6.17 MB	5.84 MB
Open Folder Main	3.51	3.50
Windows Session with Enhanced Run Mode and Private DDE and Clipboard Support		
Start PROGMAN	5.86 MB	5.69 MB
Open Folder Main	3.39	3.35

The above numbers show that how you set up your session can have a significant impact on the amount of memory used.

The next series of numbers shows how much memory is required when the particular support is is installed or activated on your system. These number are presented as Kilobytes (KB) in size. For example, Save Desktop costs 300KB in working set memory. So, if the desktop was being saved while the System folder was being opened, 3.9 MB of memory would be required instead of 3.6. The numbers below are in addition to those above.

Support	Additional Working Set Memory
Multimedia	
Device Drivers	50 KB
Desktop Folder and Objects	100
System Sounds	40 - 300 (Inactive - Active)
Save Desktop	300
HPFS	200 - 400 (No HPFS Partition Defined, and 64 K cache)
XDF Support	50
Personnel Communications 3270	
Support not started	<20
Support active	
Minimum	300 (Single session and does not
Maximum	450 include connection protocol)
TCP/IP	500 (Base support)

This is for the Device Drivers that are installed with the BonusPak.

Lan Requester 4.0	1 MB
Device Driver and Setup	700 KB
Requester Active	300

Default install for Token Ring including LAN Messaging support.

MNC 1208
Technical Coordinator Program

Bob Gress
IBM

Technical Coordinator Program

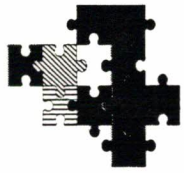
...an approach to supporting the workstation user

Bob Gress
Program Manager
Technical Coordinator Program

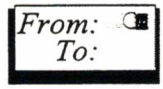
Technical Coordinator Program AGENDA

Technical Coordinator Profile

Program Components

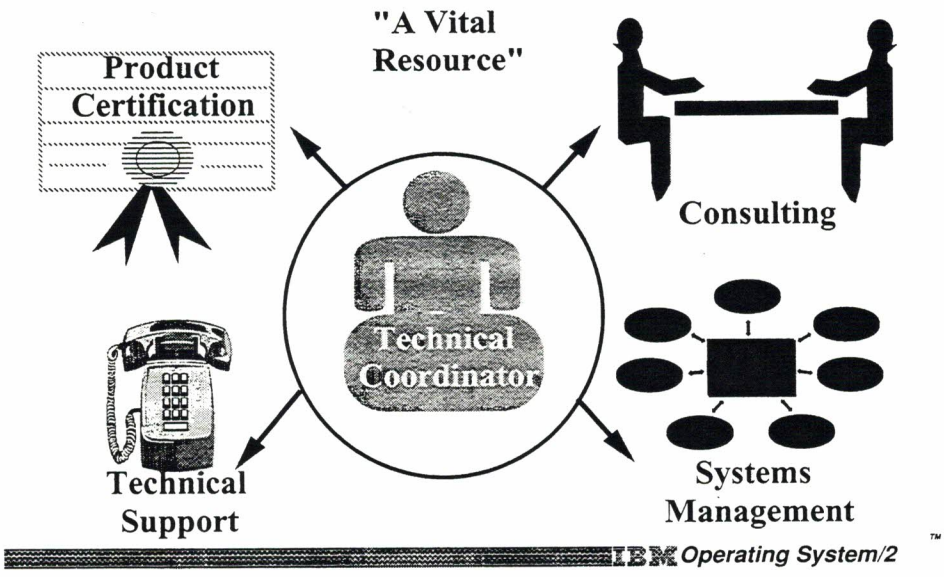


- **IBM Link**
- **OS/2 BBS**
- **Special Promotions**
- ***Personal Systems* Magazine**
- **Technical Conferences**

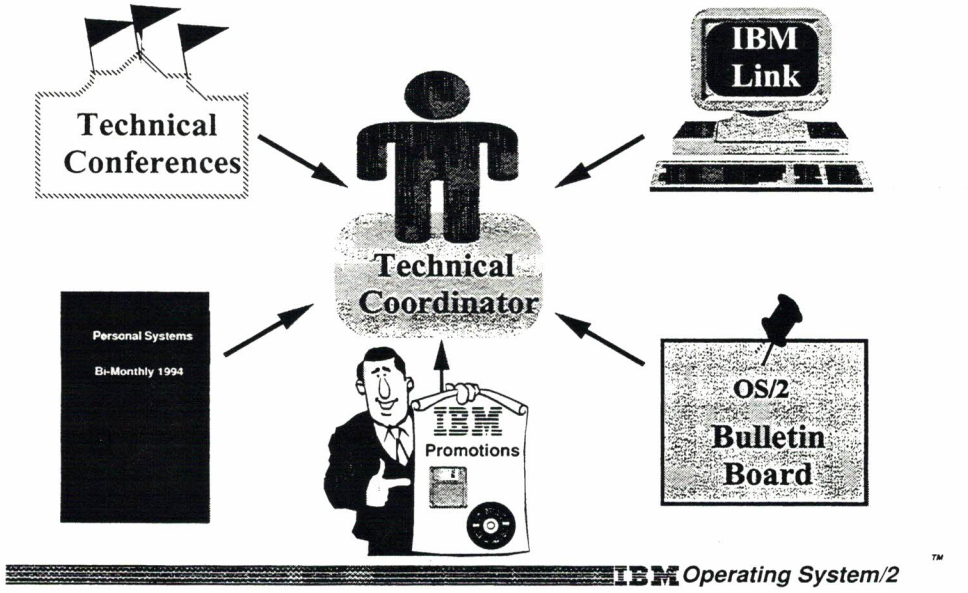


How to Enroll

Technical Coordinator Program

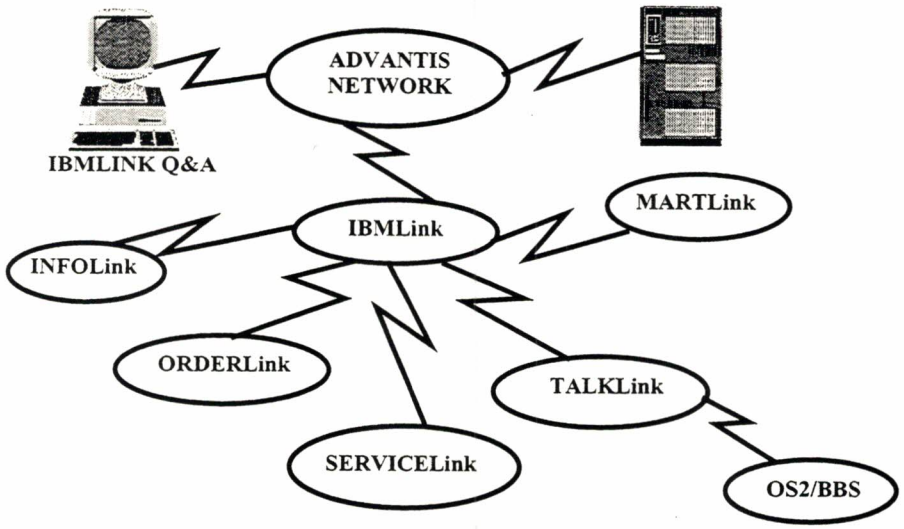


Technical Coordinator Program



Technical Coordinator Program

IBMLink Q & A



IBM Operating System/2

OS/2 Bulletin Board

- On-Line Interactive & Off-Line Batch Capabilities
  - Conferences
  - User Message Facility
  - Official Defect Support
  - Software Library - Can Download
- TECH-COR CForum
- Support
  - Available 24 Hours
    1-800-543-3912
    (IBMLink)

IBM Operating System/2

Technical Coordinator Special Promotions

- Technical Connection CDRom
- Developer Connection CDRom
- OS/2 WARP Beta
- OS/2 WARP
- Education Discounts
- Video Discounts
- Watch this Space

IBM Operating System/2™

Technical Coordinator Program

Personal Systems

IBM's Magazine for Technical Coordinators

Bi-Monthly Publication

Focus Section

- New Applications
- Customer Success Stories
- Education
- Announcements
- Product News

IBM Operating System/2™

Technical Coordinator Program

Personal Systems
IBM's Magazine for Technical Coordinators

Technical Section

- "Under the Covers" Technical Articles
- Technical Expert Authors
- IBM Developers
- Industry Consultants
- Business Partners
- Information on Emerging Technologies

Technical Coordinator Program

Personal Systems
IBM's Magazine for Technical Coordinators

Little Solutions

- Tips and Techniques
- Technical Questions & Answers
- Corrective Service Information

IBM Operating System/2 ™

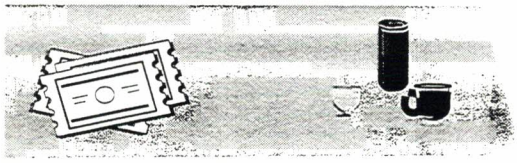
Technical Conferences

- Participate in one or two conferences each year
- TC Special Emphasis Events
  - Private Reception
  - Private Breakfasts
  - Raffles
- TC Exhibit Booth
  - Latest TC Program Information
  - "In-Put" Platform

IBM Operating System/2 ™

Technical Conferences

- Customized Welcome Packages
  - TC Tracts
  - "TC Power" T-Shirts
  - Discount coupons



IBM Operating System/2 ™

Technical Coordinator Program

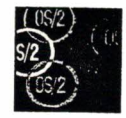
You must be registered by your IBM Marketing Rep to receive

- **IBMLink**
  - **OS/2BBS**
  - **TECH-COR CForum**
  - **ASKQ**
  - **Search Serve**
- **Personal Systems Technical Solutions**
- **Technical Conferences**
- **Technical Coordinator Special Promotions**

Phone: 1-800-547-1283

IBM Operating System/2 ™

Technical Coordinator Program



A Special Bridge

IBM Operating System/2 ™

PRO 1203
What's Wrong with Object-Oriented Programming

Bill Schindler
TGC
Cave Creek, Ariz.

What's Wrong with Object Oriented Programming

Bill Schindler

Many of the promises of object oriented programming, or OOP, sound great. But are they real, or is OOP a white rabbit leading us down a hole into a programming wonderland?

- Reusable code
- Speedier development
- Improved reliability
- Higher quality
- Easier maintenance

Hope or hype?

Does OOP fulfill these promises?

Reusable code

Code is only reusable when it is designed for reuse. To be reusable, code must be documented, accessible, and applicable (i.e. a close fit). Reuse also requires a common paradigm.

Note that none of these points apply exclusively to OOP. You could easily apply these "reuse rules" to C, COBOL, or BASIC.

In order to achieve reuse, you may need to redesign your development process. You may even need to redesign your company!

Speedier development

Speedier development is primarily achieved through reuse.

Most claims for speedier development don't offer a basis for comparison. Is OOP actually faster than your current development environment? Is OOP faster than visual programming?

Improved reliability

Again, OOP primarily achieves this promise through reuse.

OO languages use strict type checking and tend to encourage data encapsulation. But so do many modern non-OO languages!

Higher quality

This claim depends on the availabilty of tested, reusable classes.

Although the situation is improving, many companies must still develop and test their own class libraries. Creating a quality in-house class library can be equal in complexity to developing a full application.

Easier maintenance

Changes are localized and shouldn't have big side effects.

However:

- Minor changes in C++ can force a global recompile.
- Inheritance can complicate maintenance.
- Polymorphism can really complicate maintenance.
- Side effects from changes can be hard to track down in a large OO program.

So, is there any hope?

Programming problems are getting bigger and harder to solve. OOP isn't a cure-all panacea for the software industry, but it can simplify many of the hard-to-solve problems.

However, some of the first problems you'll need to confront are those of OOP itself:

- A steep learning curve
- Increased design complexity
- Slower compilation
- Increased program size
- Decreased program speed

Compilation speed, program size, and program speed are being addressed by the compiler industry. The code quality of some development environments has improved to the degree that non-OO languages now offer only a minor benefit in these areas over an OO language.

Steep learning curve

Developers must learn a new programming model and a new programming language. In many companies, the switch to OOP will force developers to confront new (to them) concepts that aren't strictly OOP: scoping, data hiding, pointers and references, threading, and so on.

Because of this, training must encompass both OOP and non-OOP concepts. Furthermore, everyone *must* be trained to evaluate, design, code, and test for reuse.

Increased design complexity

Every successful OO design is based on a view that extends well past the current application. The design must seek out reusable classes (both new and existing). The design is often complicated by the very flexibility of the OOP model.

Adopting a meta-view for design can more than double the time it takes to build an application during the early stages of converting to OOP. Experience shows a pay-off in later applications, but designers, developers, customers, and the company must be willing to wait for that pay-off.

Surviving OOP

Operating systems, applications, and development environments are all becoming object oriented. OO is going to become even more prevalent in the next few years as environments like OpenDoc and Taligent take hold.

PRO 1203 What's Wrong
Schindler, Bill

The question is no longer if you'll change to OOP, but when you'll change and whether you'll survive the change.

The first three steps you need to take to survive are:

- Get training in OOP.
- Create some small OO programs.
- Get training in OOP.

OOP is a different way of coding, designing, and even thinking about software.

Next, seek out tools that increase OOP's power and simplify your job by offering you instant reuse benefits:

- SOM (System Object Model), DSOM, and Direct-to-SOM compilers
- OpenDoc
- Workplace Shell
- Taligent
- VisualAge

Summary

OOP is surrounded by hype. Most of the hype is based on reuse, which is not magically granted through the use of an OO language. Furthermore, OOP has a steep learning curve and may add complexity to early OO design work.

OOP itself is a powerful programming tool. Like any power tool in the wrong (i.e. untrained) hands, it can wreak havoc and destruction. To use OOP, you must learn a new model for coding and designing software.

To survive a change to OOP: Get trained in OOP. Look for tools and environments that give you the benefits of reuse.

PRO 1204
Agency: Mobile Agents in Object REXX

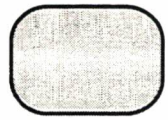
Rick McGuire
IBM

"The Agency": An Object REXX Mobile Agent Prototype

Rick McGuire
Object REXX Project
rick_mcguire@vnet.ibm.com

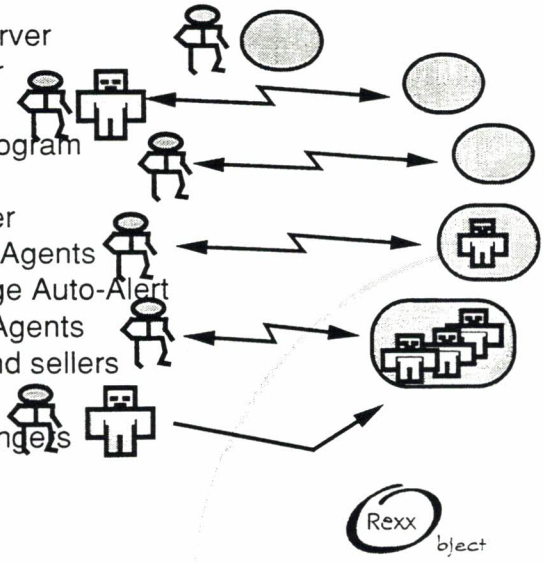
Communications Modes for "Mobile Computing"

- Client Programs
  - Used directly by users
  - Always local
- Server Programs
  - Invoked by client programs
  - May be local or remote
- Agent Programs
  - Work independently for users, even when not connected



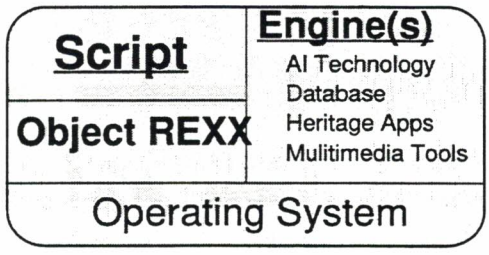
"Mobile Computing": Modes of Communications

- Local Application/Server
  - My word processor
- Local Agent
  - My mail filtering program
- Remote Server
  - My database server
- Remote Server with Agents
  - My Stock Brokerage Auto-Alert
- Remote Interactive Agents
  - Brokers, buyers and sellers
- Wandering Agents
  - Information Scavengers



Rexx object

A Model for Intelligent Agents



Interface Scripts

- Tailored to users and jobs
- Simpifies using the Engines
- Portable across OSs, HW
- Integrates engines around a task

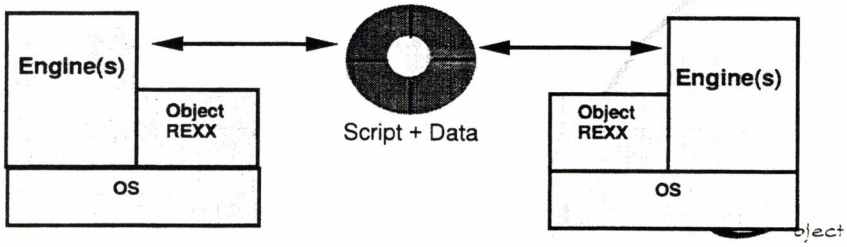
Processing "Engines"

- Often compiled, HW-specific
- Often using AI techniques
- Complex, arcane, big, fat, ugly
- Often multiple engines/solution

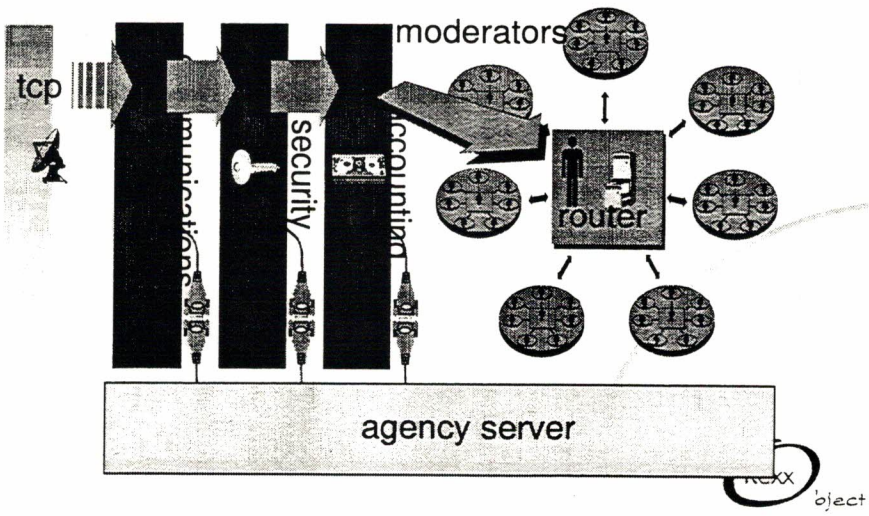
Rexx object

▼ *Mobile* Agents

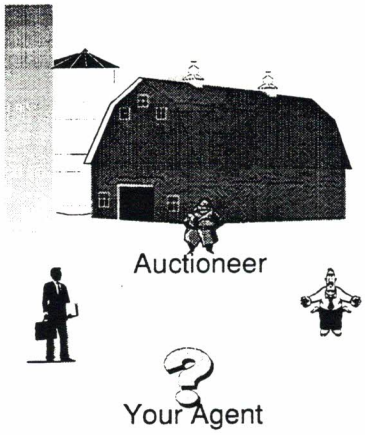
- If your solution needs to run on another system...
  - Moving "engines" around is impractical.
  - Scripts move between systems more easily.
    - ► *Physically smaller than "engine" programs*
    - ► *Interpreted languages not HW- or OS-specific*
    - ► *Move state data + script = an Object*



▼ "The Agency" Prototype



Moyle's Auction House



Typical Agent-based solutions

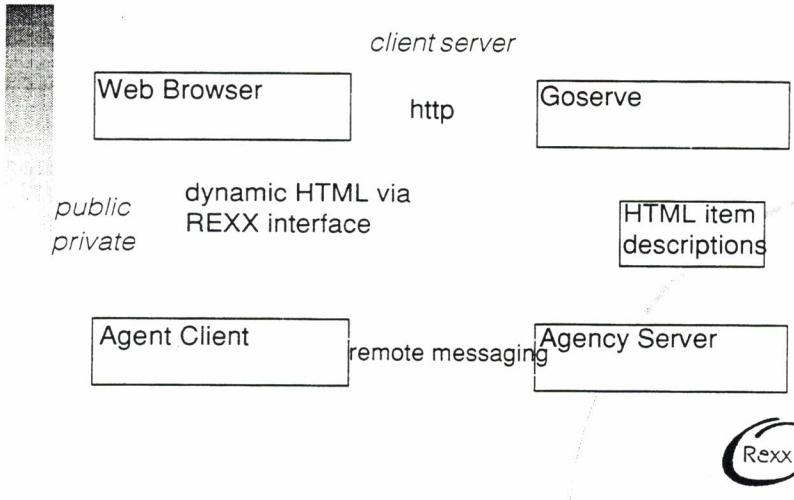
- Agents work on Users' behalf
- Agents are mobile over TCP/IP
- Agents carry instructions (shopping list and max bids)
- Agents interact at meeting place (bidding process)
- Moderator agent facilitates interaction of other agents

Object REXX scripts coordinate multiple other programs

- TCP/IP
- HTTP Server (GoServe)
- OS/2 Web Browser

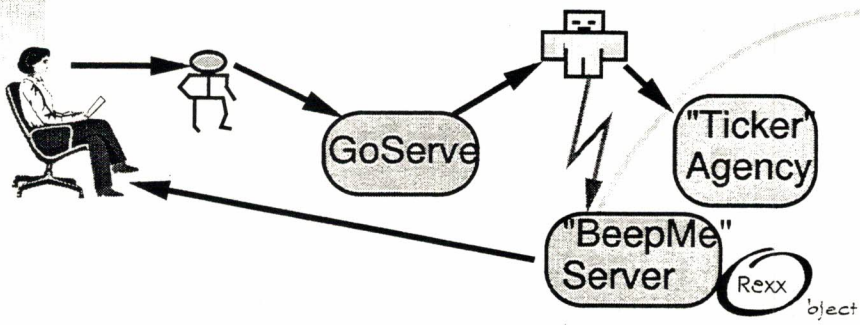
Rexx object

Private and Public Interfaces



"Ticker": Agents as Clients

User commissions Agent via dynamic HTML page
At Agency, agent enrolls with "Ticker" moderator
Agent 'watches' stock ticker datastream
If Agent sees 'alert' event, calls TCP 'Beeper' service
Beeper service alerts user



Object Rexx and Agents

- Object REXX is well-suited for scripting agents:
  - Rich function for scripting other programs
    - *Access to SOM, OpenDoc classes*
    - *Many products have REXX interfaces today*
  - Concurrent objects = real-world modelling
  - Across the important platforms
  - Free on most, cheap on others
    - *OS/2, AIX, MVS, VM*
    - *Unix source available*

Rexx bject

SYS 1205
OS/2 Systems Management Toolbox

Craig Elliott
IBM

IBM Solutions for a small planet.

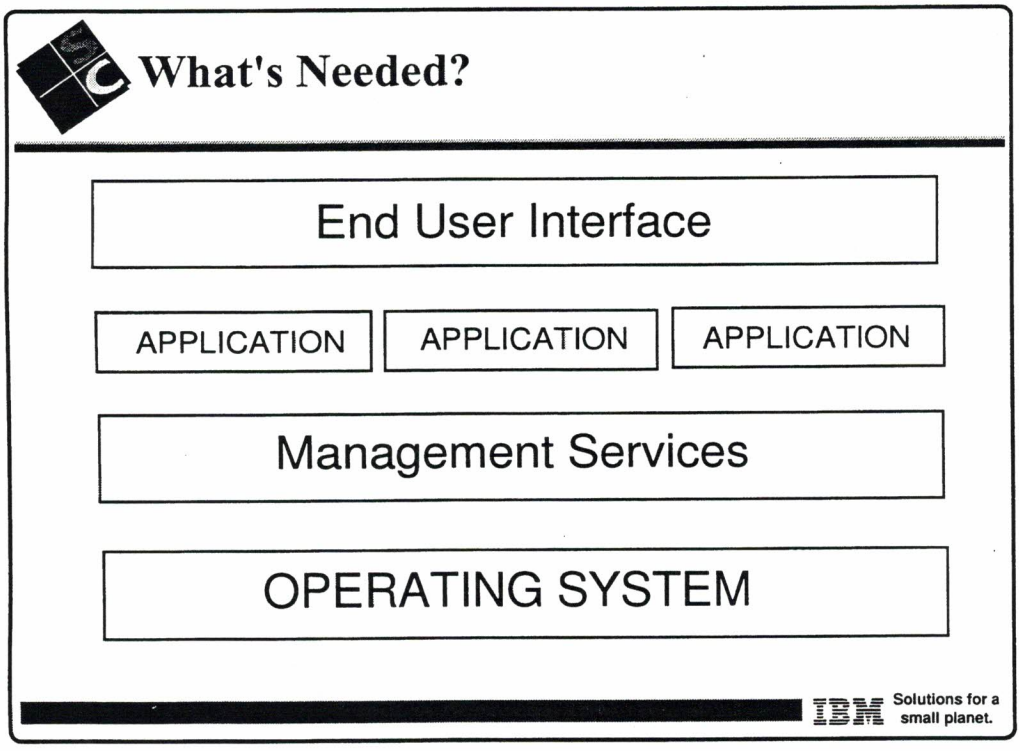
OS/2 Systems Management Toolbox

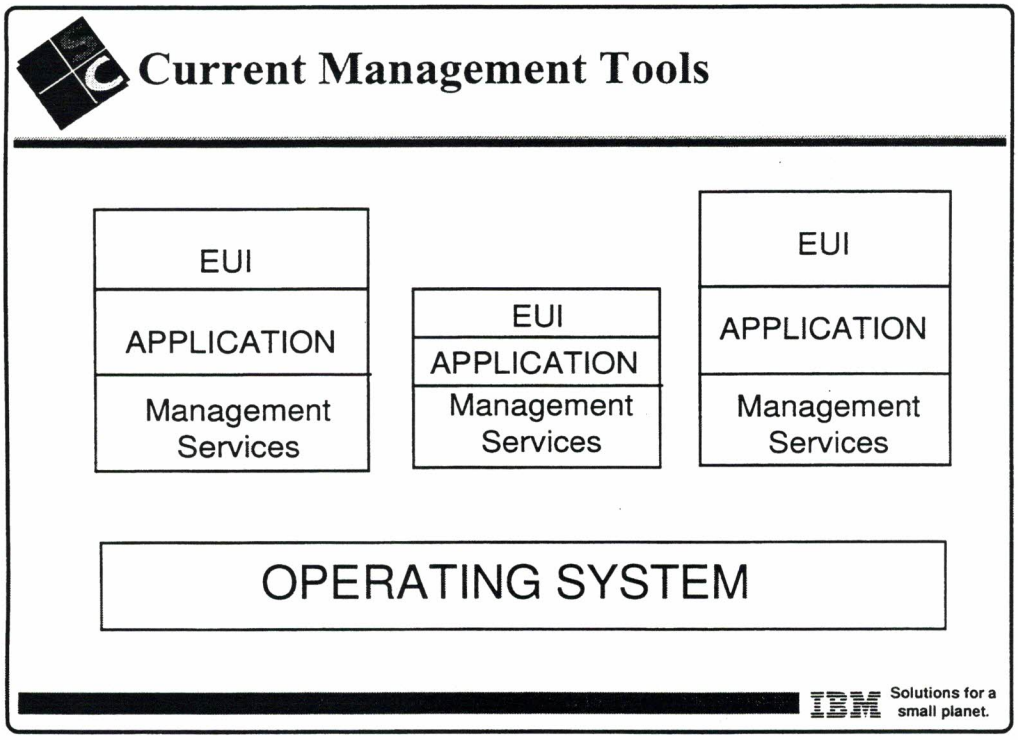
Presented by:
Craig Elliott

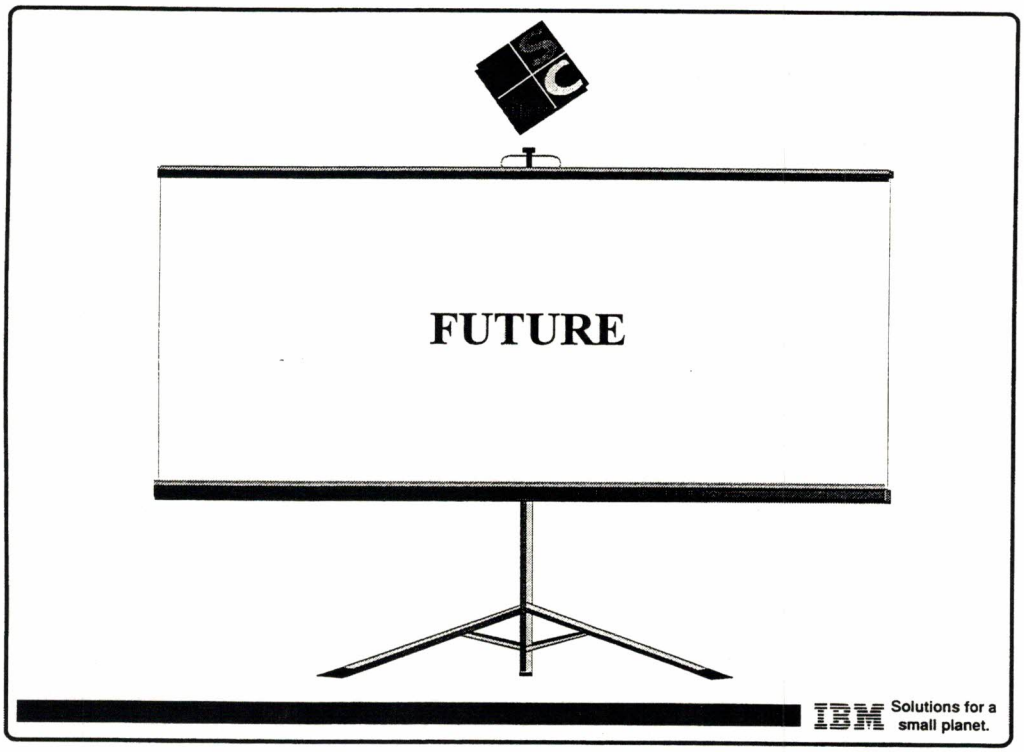
Agenda

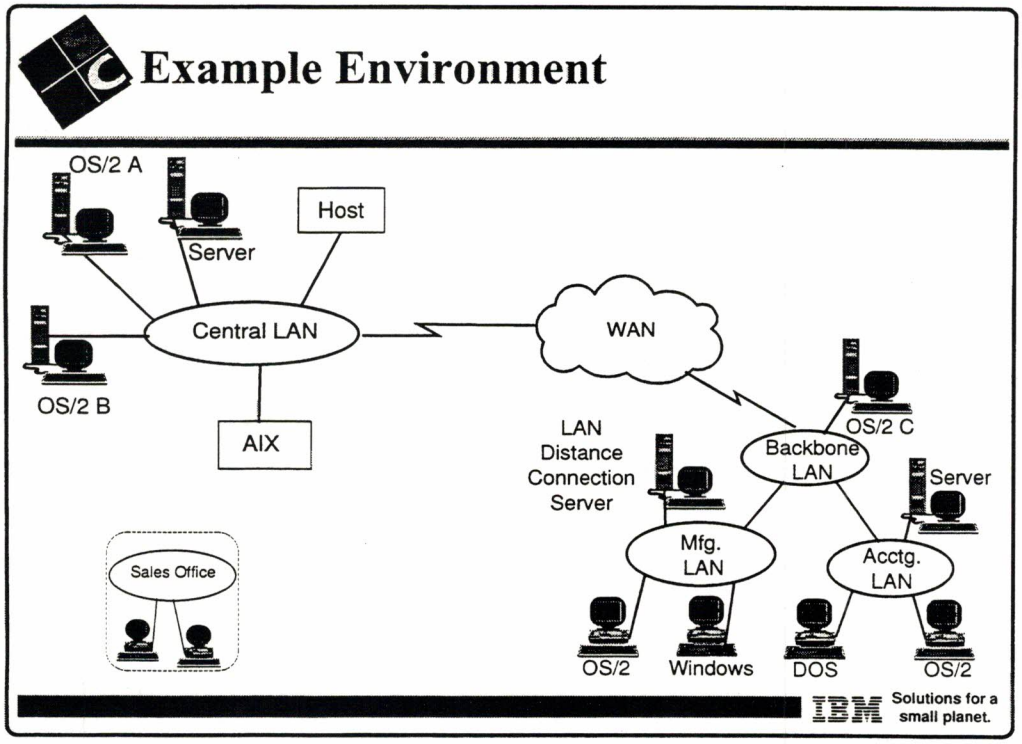
Software Distribution
Remote Operations
Hardware Inventory
Performance Monitoring
Fault Management
Backup/Recovery

IBM Solutions for a small planet.


What's Needed?
End User Interface
APPLICATION
APPLICATION
APPLICATION
Management Services
OPERATING SYSTEM
IBM Solutions for a small planet.


Current Management Tools
EUI
APPLICATION
Management Services
EUI
APPLICATION
Management Services
EUI
APPLICATION
Management Services
OPERATING SYSTEM
IBM Solutions for a small planet.


FUTURE
IBM
Solutions for a small planet.


Example Environment
OS/2 A
Server
Host
Central LAN
OS/2 B
AIX
WAN
LAN Distance Connection Server
OS/2 C
Backbone LAN
Server
Mfg. LAN
Acctg. LAN
Sales Office
OS/2
Windows
DOS
OS/2
IBM
Solutions for a small planet.

ADSM/2 - Server

Manages data backed up or archived

Multitasking capability

Supports DASD, optical disk and tape

Optionally consolidates data from each individual workstation to minimize retrieval time.

IBM Solutions for a small planet.

ADSM/2 - Administrative Client

GUI for company wide backup/restore or archive/retrieve

Establish consistent storage management policies across the enterprise

Centrally schedule backup and archive

IBM Solutions for a small planet.

ADSM/2 - Backup-Archive Client

GUI for backup/restore or archive/retrieve

Optional compression

Allows restore and retrieve files onto different client nodes

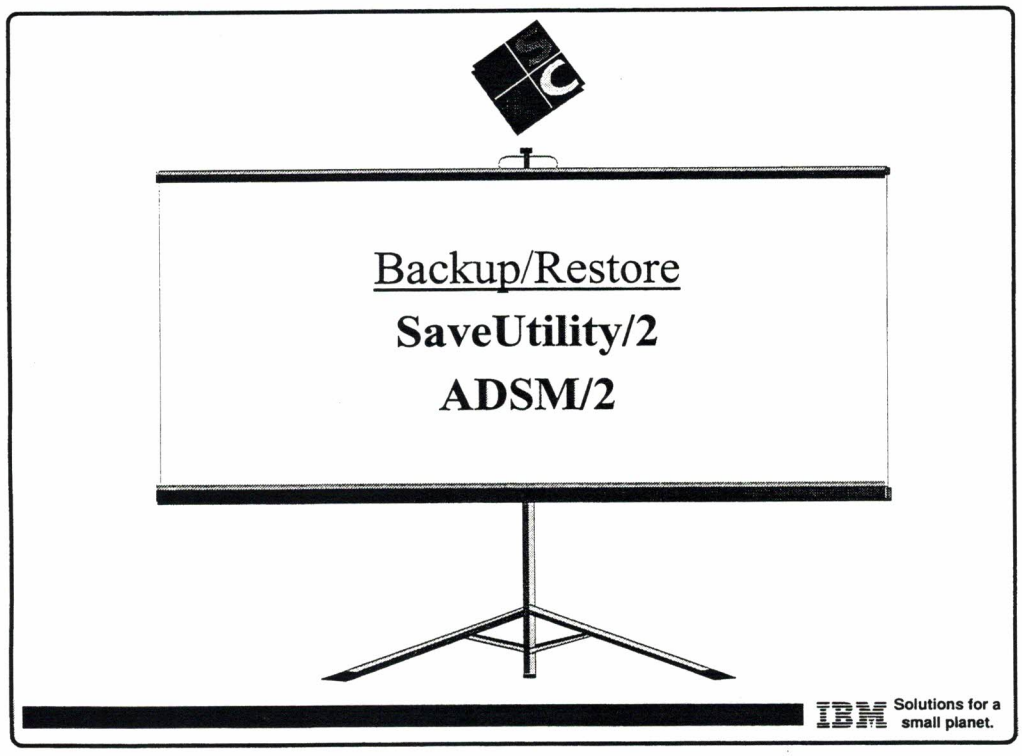
Grant authority to other users to access specific files that have been backed up or archived

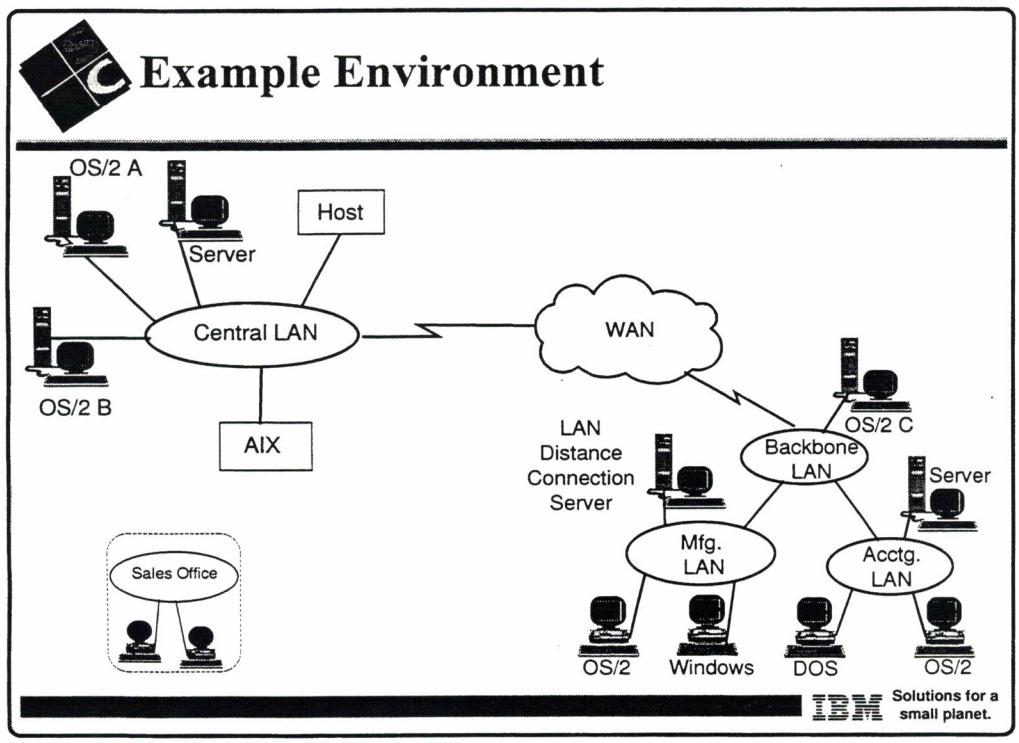
IBM Solutions for a small planet.

SaveUility/2 Features

Full OS/2 file and operating system backup
Full DOS file and operating system backup
Provides management and administration
File, subdirectory and disk restore
Full system restore from period of last backup
Various archive device support
Support up to 250 workstations
Operates with DOS, OS/2, Windows and Novell

IBM Solutions for a small planet.


Backup/Restore
SaveUtility/2
ADSM/2
IBM Solutions for a small planet.


Example Environment
OS/2 A
Server
Host
Central LAN
OS/2 B
AIX
WAN
OS/2 C
LAN
Distance
Connection
Server
Backbone
LAN
Server
Mfg.
LAN
Acctg.
LAN
Sales Office
OS/2
Windows
DOS
OS/2
IBM Solutions for a small planet.

NetFinity Alert Actions

Automatic pager dial-out

Send a SNMP version of the alert

Issue pop-up message

Launch user defined program

Play .WAV files on Multimedia enabled systems

IBM Solutions for a small planet.

NetFinity Alert Manager Service

Generate and forward alerts
Alerts generated locally and remotely
Offers a large standard set of alerts
Allows a customer to supply a custom alert handler
Maintains a log of all alerts

- Specifies alert action conditions

- Severity, type, application ID, sender ID, application type ID

IBM Solutions for a small planet.

NetView for OS/2

System
- Generic trap 0 - 5

- Operating System
  - Average percent CPU in idle mode over the interval
- LAN Requester
  - Cannot get name or number of machine id
  - Reached the configured limit
- LAN Server
  - Server unable to contact primary domain controller
  - Detected multiple failed password attempts

IBM Solutions for a small planet.

NetView for OS/2

Issue messages without requests

Receive event notifications
- Errors
- Changes

- Process received events
  - Log
  - Display
  - Automation

IBM Solutions for a small planet.

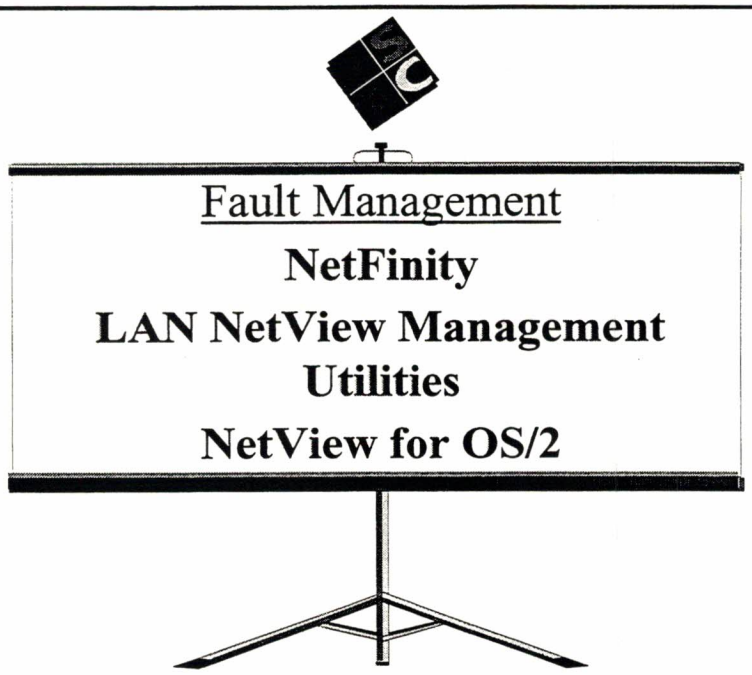
LMU Fault Management

Intercepts and responds to alerts
Alert forwarding capabilities
Alerts

- Workstation and server performance
- Application and virus detection
- NetBIOS and I PX / SPX protocol statistics
- Disk volume

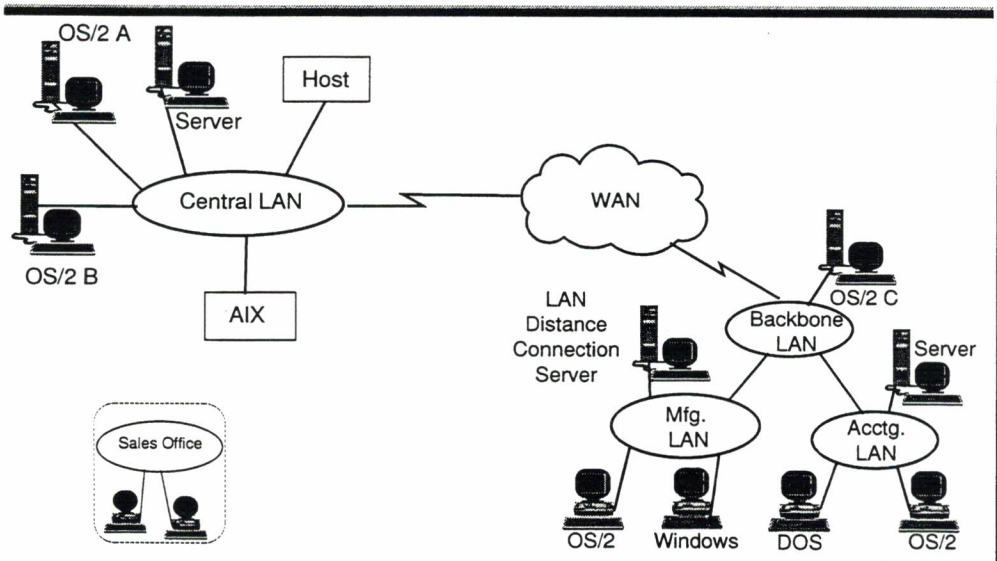
- Heartbeat

IBM Solutions for a small planet.



IBM Solutions for a small planet.

Example Environment



IBM Solutions for a small planet.

NetFinity Monitoring Service

Continuous system monitoring

- Locked and virtual memory
- CPU usage
- DASD space and utilization
- Read/write errors (NetFinity Manager system only)

- Line graph, text or real-time views
- Unique data handling technique to conserve disk space over long periods of time
- Threshold alert generation both for level exception and/or time duration

IBM Solutions for a small planet.

NetView for OS/2

System
- CPU utilization, memory used, etc.

- Operating System
  - Software running, software memory usage, software CPU usage, etc.

- LAN Requester
  - Bytes/second read/write, buffer allocation failure, etc.

- LAN Server
  - Server response time, buffer requests, heap space, etc.

IBM Solutions for a small planet.

NetView for OS/2

Gather performance data
- CPU Utilization
- Memory usage...

- Watch for performance problems
  - Thresholds
  - Traps

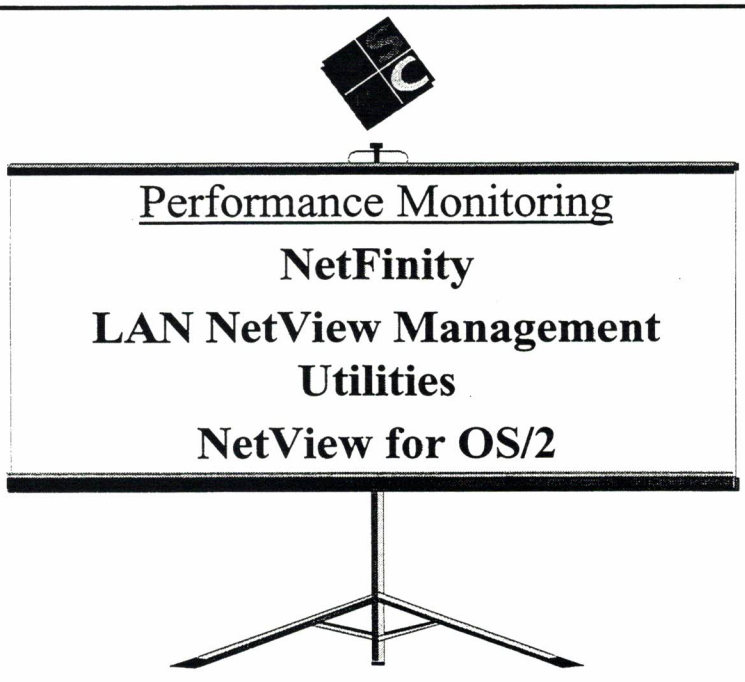
- Plan for resource enhancements
  - Trending analysis

IBM Solutions for a small planet.

LMU Performance Management

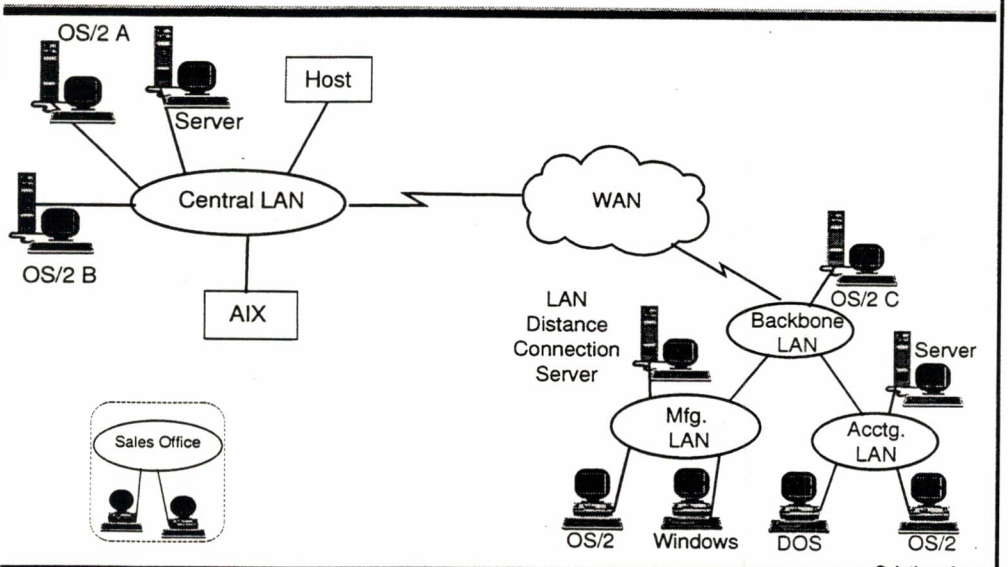
Collects resource performance data
Collects adapter status and ring utilization
Collects LAN Server statistics
Monitors IPX and SPX protocol statistics
Collects disk volumes statistics
Collects CPU statistics

IBM Solutions for a small planet.



IBM Solutions for a small planet.

Example Environment



IBM Solutions for a small planet.

NetFinity

	All Machines	IBM PS/2
- Processor		
- Memory amounts and address ranges		
- Disk subsystem		
- Planar video subsystem and display		
- Peripherals (keyboard, mouse, I/O ports)		
- Operating system		
- Performance monitoring (CPU, disk, memory)		
- Alerting		
- Remote session		
- Remote screen view		
- File transfer		
- User profile		
- Adapters		
- Serial numbers		
- C2 security features		
- Error log analysis		
- Enhanced memory subsystem		
- Non-planar video subsystem		
- Full model/submodel identification		
- Reference partition (IML) support		
- ECC memory		
- RAID 5 subsystem		

IBM Solutions for a small planet.

LMU - Sample VPD (cont.)

User Data

Critical File Information

```
C:\CONFIG.SYS
     Location              C:\
     Date                  1995-03-17
     Time                  15.41.48
     Size                  4745 bytes

C:\STARTUP.CMD
     Location              C:\
     Date                  1995-02-24
     Time                  10.35.16
     Size                  230 bytes

FLG.EXE
     Location              E:\FLG\
     Date                  1995-02-13
     Time                  16.38.16
     Size                  81967 bytes
```

IBM Solutions for a small planet.

LMU - Sample VPD (cont.)

```
Keyboard Type ..... 101/102 Key Enhanced (ID AB83)

Current Display ..... PS/2 Color 8512/8513/8515

Primary Node Address ... 10005A8FAFA0
Primary Universal ..... 10005A8FAFA0


Total Slots ............ 8
Expansion Slot 1 ....... IBM PS/2 SCSI Adapter with Cache
  Status                 Enabled
Expansion Slot 2 ....... IBM 16/4Mb Token Ring Adapter/A
  Status                 Enabled
Expansion Slot 3 ....... IBM Multi-Protocol Communications Adapter
  Status                 Enabled
Expansion Slot 5 ....... IBM XGA-2 Display Adapter
  Status                 Enabled
```

Logical Drives

```
FAT     Drive C ........   41536 of  102166 KB free ==>  59% full
FAT     Drive D ........  308672 of  689984 KB free ==>  55% full
```

IBM Solutions for a small planet.

LMU - Sample VPD (cont.)

```
                    Hardware Configuration

Machine Type ..... IBM PS/2 Model 95 XP
Product Number ..... 8595-xxx
Processor ..... Intel
  Processor Speed        66 MHz
CoProcessor ..... Intel 80486
Bus Type ..... Micro Channel 32-Bit
Total Memory ..... 65152 KB = 63.6 MB
Equipment List ..... 1 Parallel Port(s)
                     1 Serial Port(s)
                     2 Diskette Drive(s)
                     3 Fixed Disk(s)
                     Math CoProcessor
Serial Port 1 ..... COM1
  Baud rate           1200 bps
  Data bits           7
  Parity              Even
  Stop bits           1
Diskette Drive 1 ..... 3.50"  - 2949K - 80 Tracks - Type 6
Diskette Drive 2 ..... 3.50"  - 1474K - 80 Tracks - Type 4
Fixed Disk 1 .....  517 MB  =  529408 KB  =  542113792 bytes
Fixed Disk 2 .....  958 MB  =  980992 KB  = 1004535808 bytes
Fixed Disk 3 .....  382 MB  =  391168 KB  =  400556032 bytes
```

IBM Solutions for a small planet.

LMU - Sample VPD

```
IBM LAN NetView Management Utilities
5622-153 (C) Copyright IBM Corp. 1991, 1993. All rights reserved.

IBM LAN NetView Management Utilities Maintenance Level LM00200

                    Report Generation

  Date of report ..... 1995-04-04
  Time of report ..... 11.04.37

                  Operating Environment

  Operating System ..... OS/2 3.00
```

IBM Solutions for a small planet.

LMU

Query Vital Product Data
User defined VPD
User defined critical file list
User defined microchannel adapter list

IBM Solutions for a small planet.

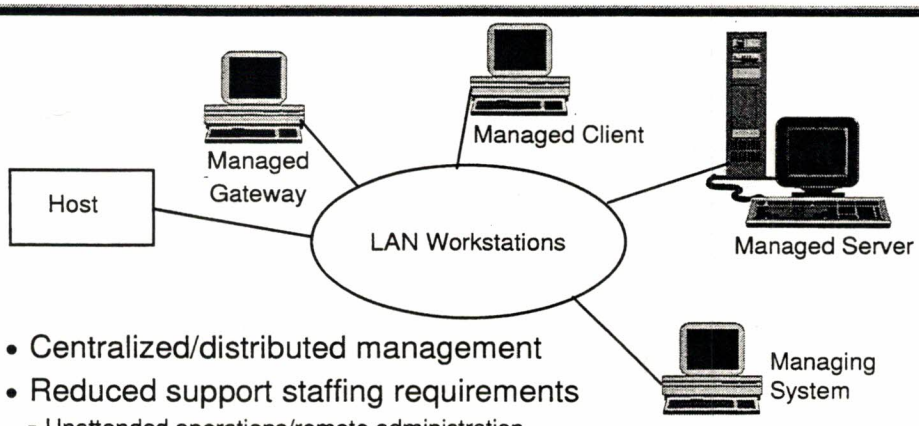
Hardware Inventory

NetFinity

LAN NetView Management Utilities

IBM Solutions for a small planet.

Systems Management: Key Requirements



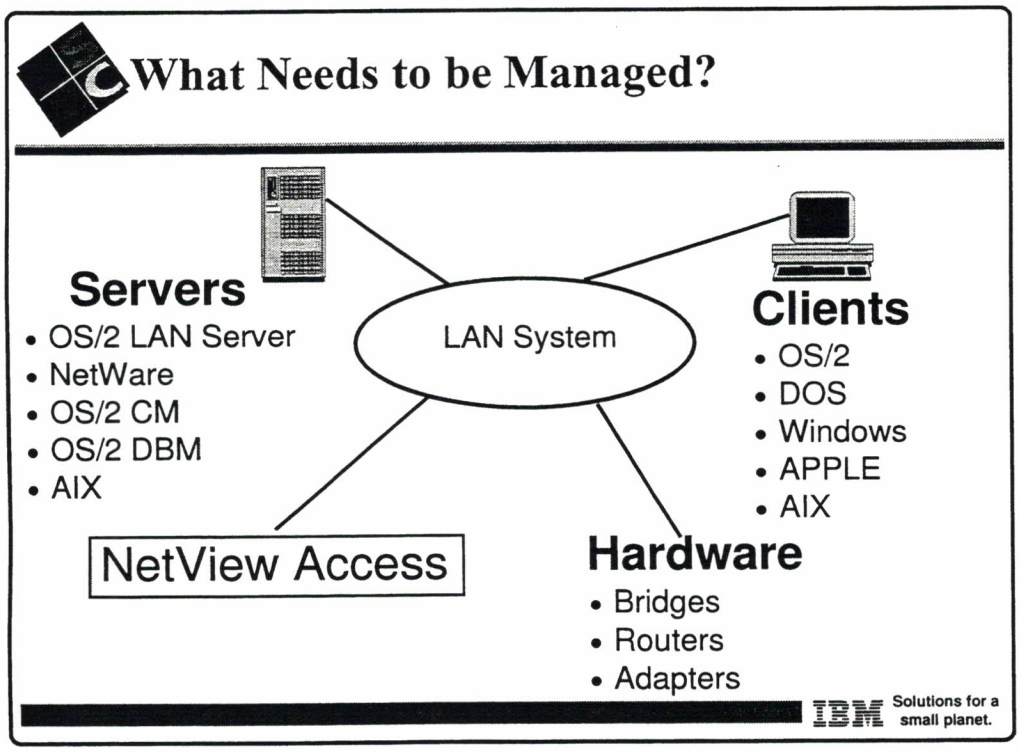
- Centralized/distributed management
- Reduced support staffing requirements
  - Unattended operations/remote administration
  - Remote software distribution/maintenance
  - Automated fault recovery
- Open system

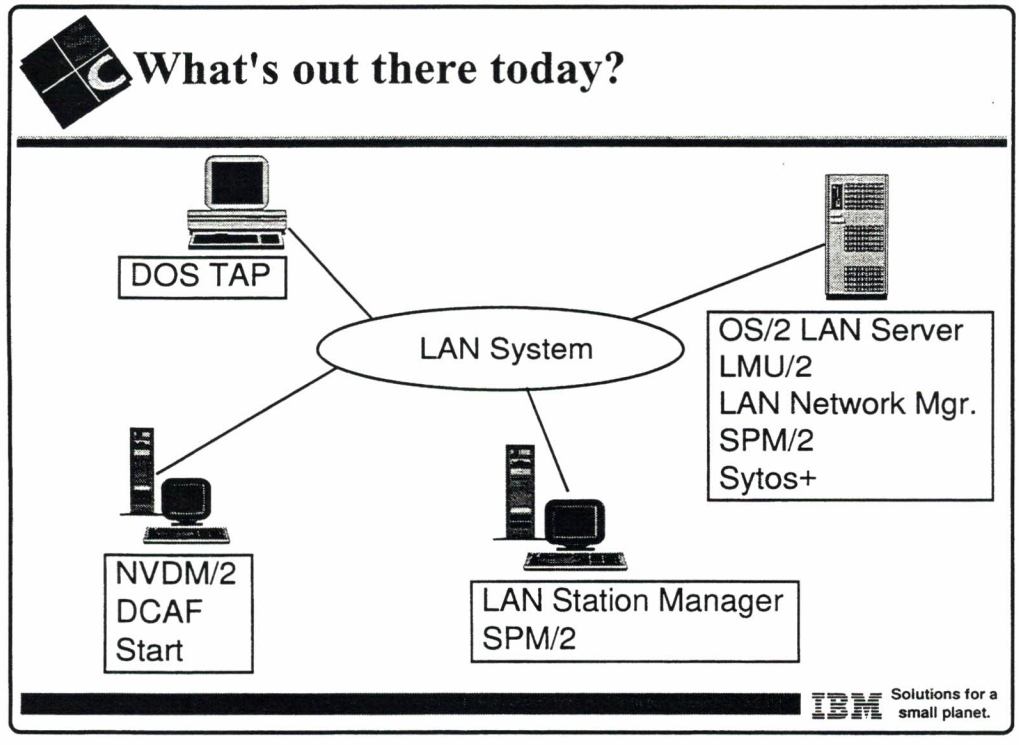
IBM Solutions for a small planet.

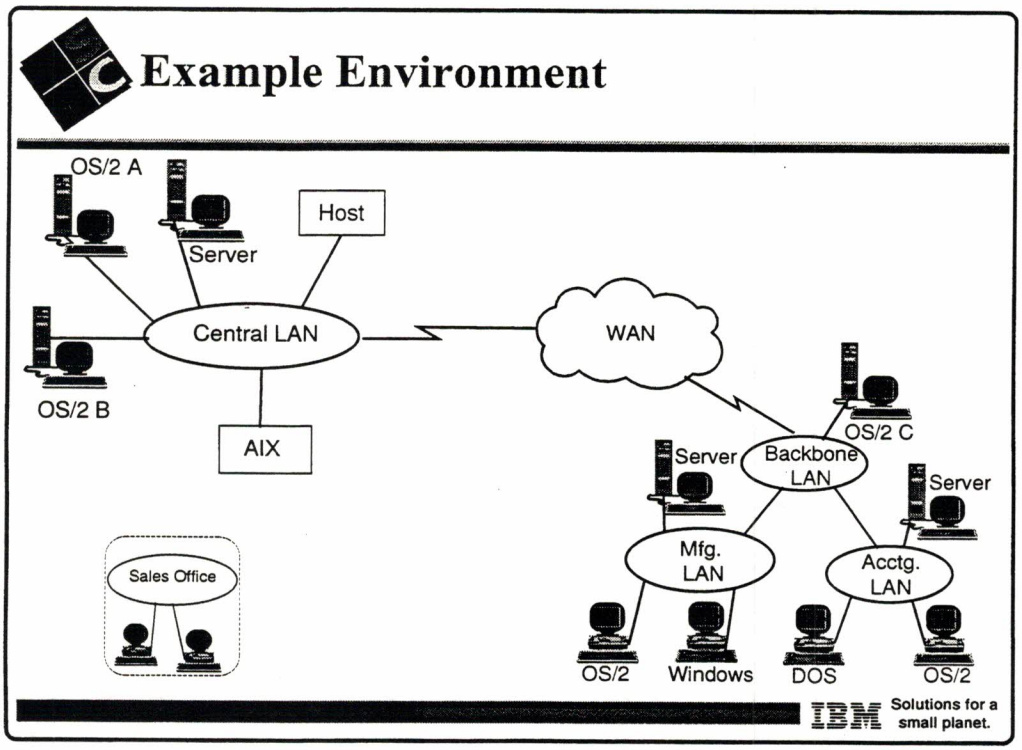
Questions

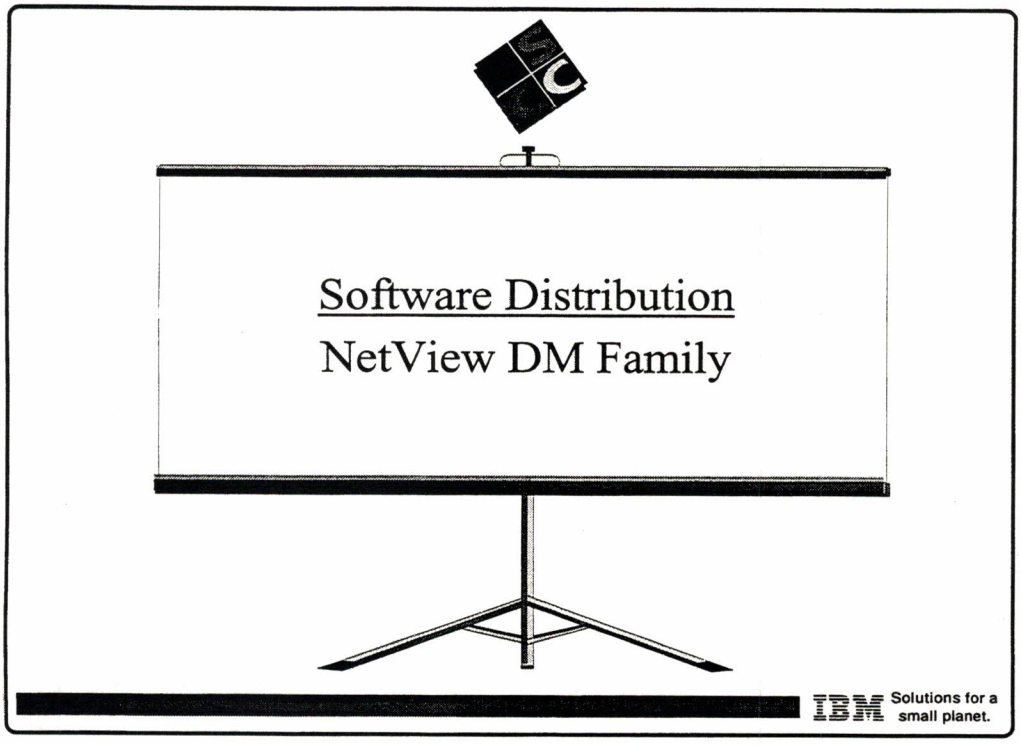
What hardware/software is installed?
How do I identify and fix problems?
Where is it?
Who's using what and how is it being used?
When do I need to install more?
How big of a support staff do I need?
How...

IBM Solutions for a small planet.


What Needs to be Managed?
Servers
• OS/2 LAN Server
• NetWare
• OS/2 CM
• OS/2 DBM
• AIX
LAN System
Clients
• OS/2
• DOS
• Windows
• APPLE
• AIX
NetView Access
Hardware
• Bridges
• Routers
• Adapters
IBM Solutions for a small planet.


What's out there today?
DOS TAP
LAN System
OS/2 LAN Server
LMU/2
LAN Network Mgr.
SPM/2
Sytos+
NVDM/2
DCAF
Start
LAN Station Manager
SPM/2
IBM Solutions for a small planet.


Example Environment
OS/2 A
Server
Host
Central LAN
OS/2 B
AIX
WAN
OS/2 C
Server
Backbone LAN
Server
Mfg. LAN
Acctg. LAN
Sales Office
OS/2
Windows
DOS
OS/2
IBM
Solutions for a small planet.


Software Distribution
NetView DM Family
IBM
Solutions for a small planet.

Customer Requirements

Eliminate "SneakerNet" as the distribution process
Eliminate diskettes at the workstation
Eliminate the requirement for skilled operators at each workstation
Reduce requirement for replication
Provide for centralized management of the software distribution process

IBM Solutions for a small planet.

Configuration Installation Distribution

Eliminate end user involvement
- Redirection: No Diskettes
- Response file: No end-user responses
- Execute install at target

- Automate distribution
  - Change management focal point
  - Profiling/dynamic grouping
  - Distribution server/client
- Consolidate configuration administration
  - Topology database
  - Response file creation

IBM Solutions for a small planet.

NetView DM/2

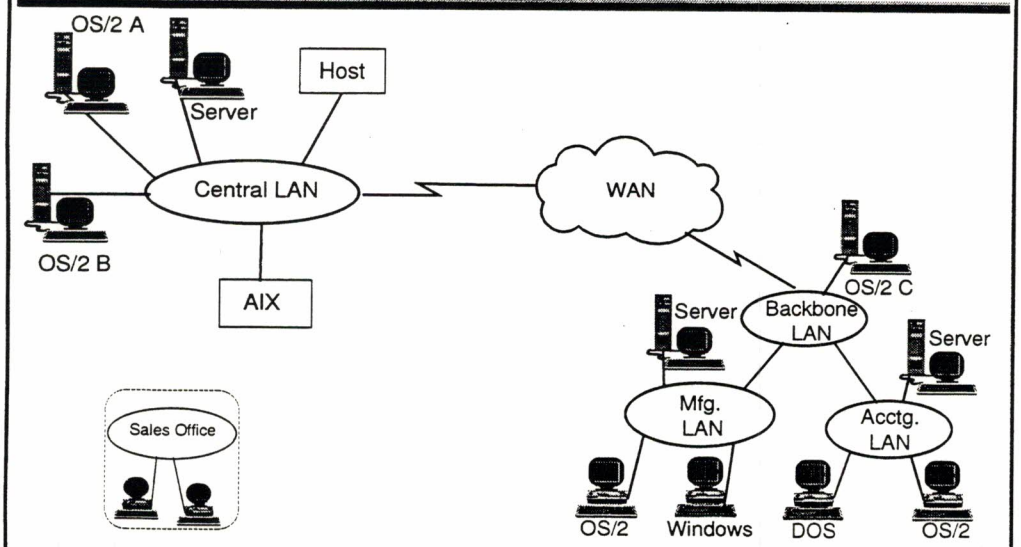
Change management focal points

- Originator of change management actions
- Connects via WAN to CC Servers
- Maintains history

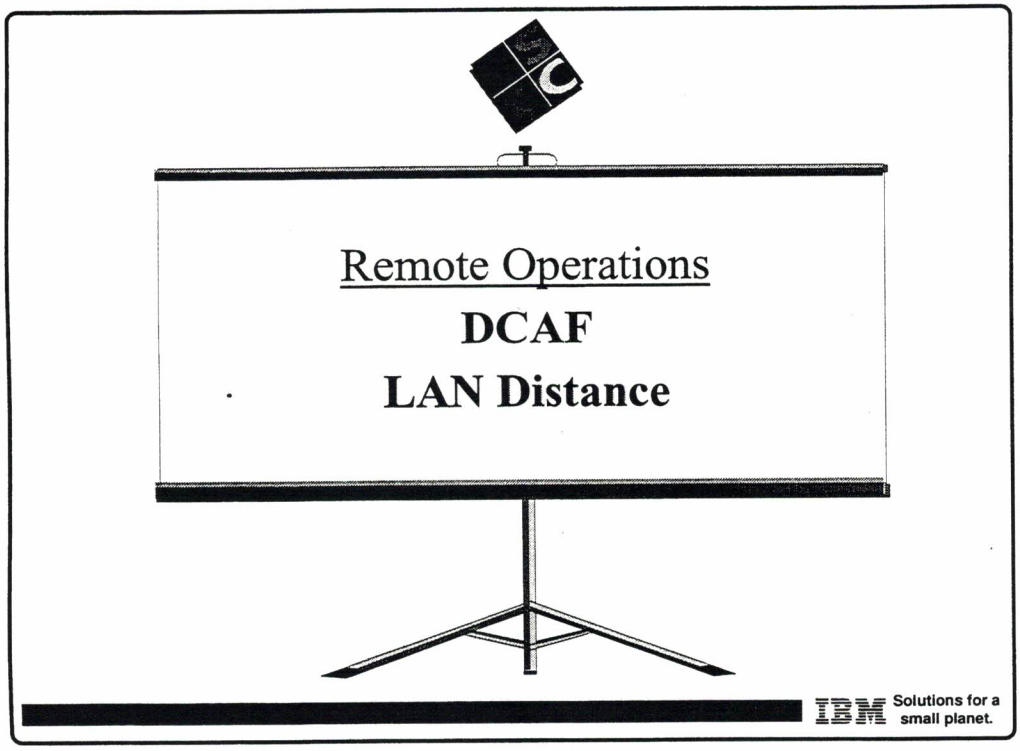
- **Change Control(CC) Servers**
  - Both target and originator of management actions
  - Maintains history
- **Change Control(CC) Clients**
  - Target of change management actions
  - Must connect via LAN to CC Server

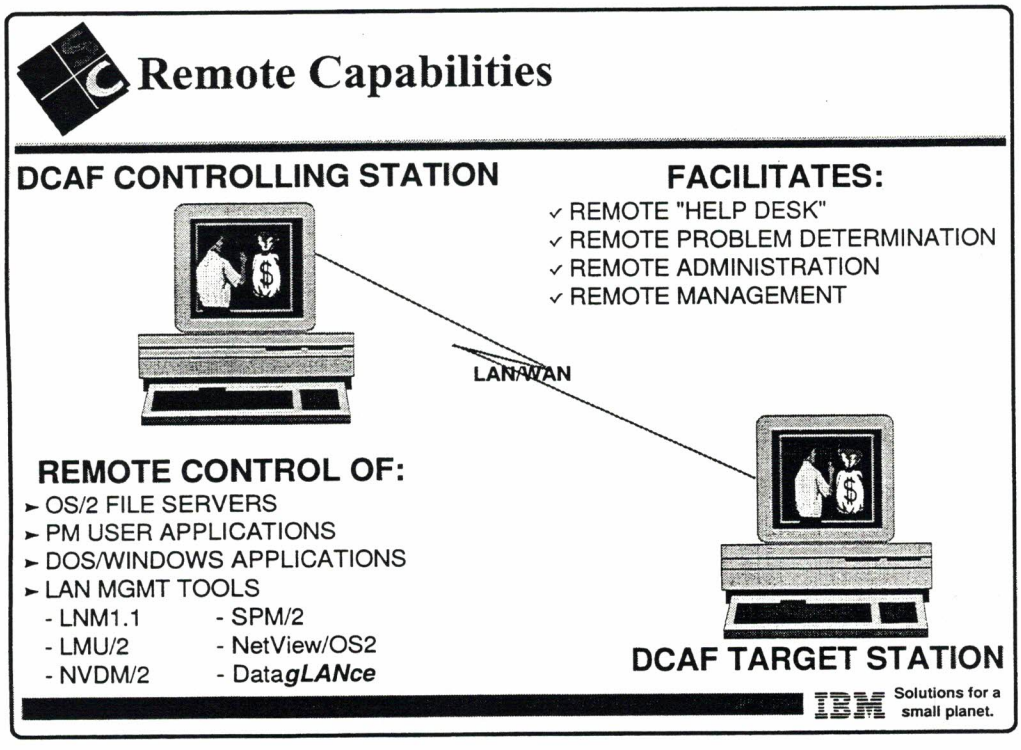
IBM Solutions for a small planet.

Example Environment



IBM Solutions for a small planet.


Remote Operations
DCAF
LAN Distance
IBM Solutions for a small planet.

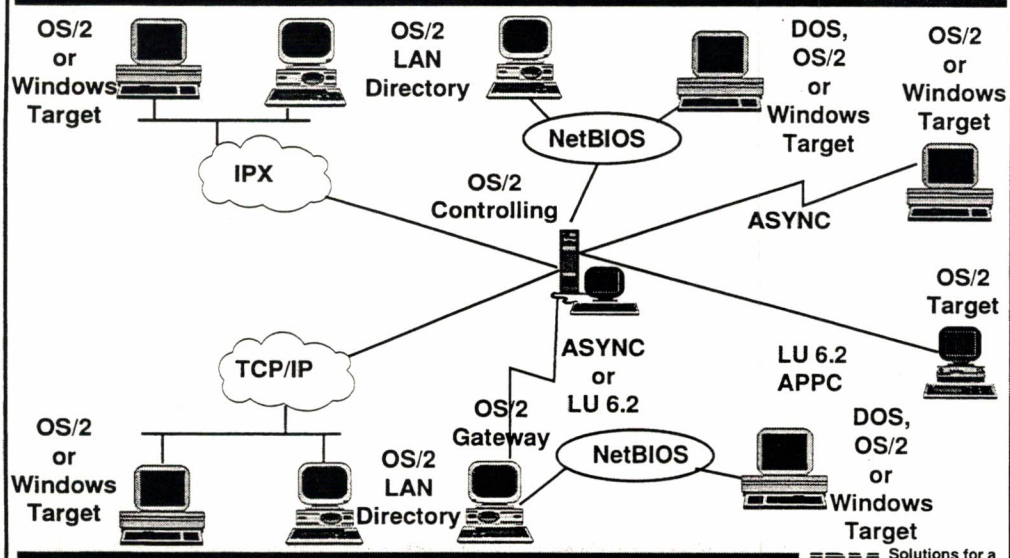

Remote Capabilities
DCAF CONTROLLING STATION
FACILITATES:
✓ REMOTE "HELP DESK"
✓ REMOTE PROBLEM DETERMINATION
✓ REMOTE ADMINISTRATION
✓ REMOTE MANAGEMENT
LAN/WAN
REMOTE CONTROL OF:
➢ OS/2 FILE SERVERS
➢ PM USER APPLICATIONS
➢ DOS/WINDOWS APPLICATIONS
➢ LAN MGMT TOOLS
- LNM1.1 - SPM/2
- LMU/2 - NetView/OS2
- NVDM/2 - DataGLANce
DCAF TARGET STATION
IBM Solutions for a small planet.

Functional Capabilities

- Remote control capacity
- Fullscreen interactive
- Active and monitor modes
- Bi-directional file transfer
- One-to-many capability
- Multiple security levels
- Program API
- On-line help and users guide
- C/I/D Enabled

IBM Solutions for a small planet.

Connectivity Options



IBM Solutions for a small planet.

Controlling DCAF

OS/2 PWS that initiates session
Select link record for target
Can switch active/monitor
Can initiate file transfer
- OS/2 to OS/2, DOS, or DOS/Windows

- Can reboot target PWS
- One-to-many support

IBM Solutions for a small planet.

Gateway and Directory

GATEWAY
- Performs protocol conversion
- Multiple simultaneous sessions
- Audit trail for security
- Fast-path access optional

- DIRECTORY
  - Targets are based on UAA or LAA with optional Alias
  - Directory shows target status
  - Unlimited Directory capacity
  - Direct IP/IPX/NETBIOS access or via Gateway

IBM Solutions for a small planet.

Target

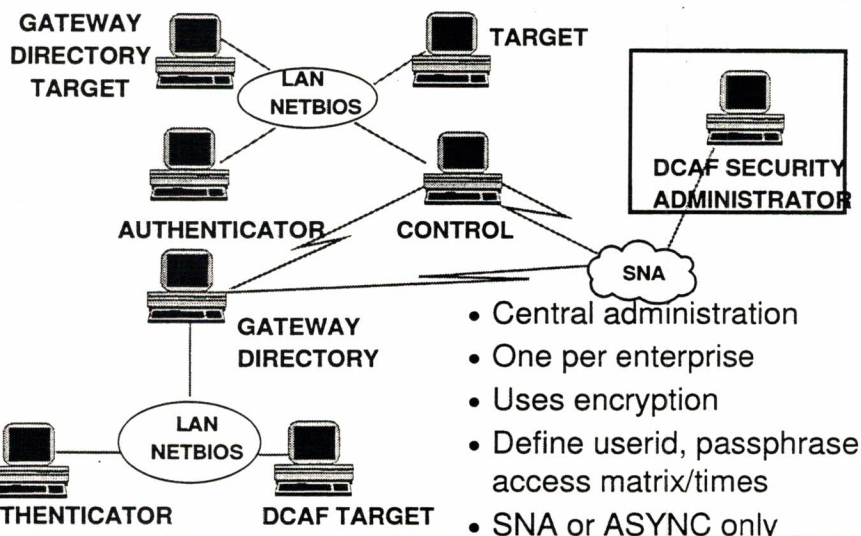
Target types:

- OS/2
- DOS
- DOS/Windows

- Active/monitor modes
- Bi-directional file transfer
- Hot key control
- Security levels:
  - None
  - Password
  - Full encryption

IBM Solutions for a small planet.

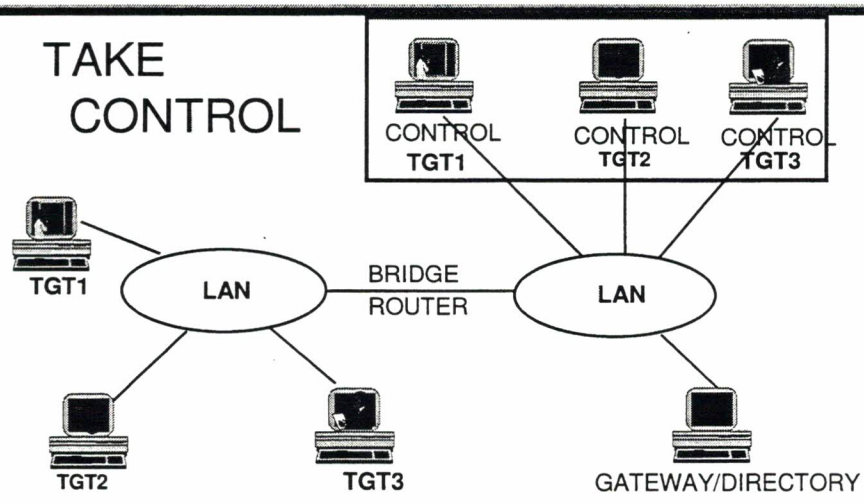
Security - Administrator



- Central administration
- One per enterprise
- Uses encryption
- Define userid, passphrase, access matrix/times
- SNA or ASYNC only

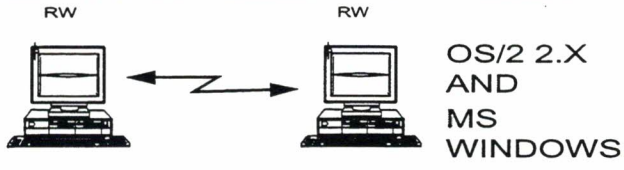
IBM Solutions for a small planet.

Help Desk



IBM Solutions for a small planet.

LAN DISTANCE REMOTE



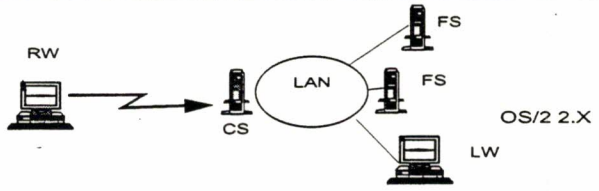
Enables remote workstations to dial-in and operate as if LAN attached

Enables creation of a virtual LAN for point-to-point environments

Two concurrent connections supports creation of virtual networks(OS/2)

IBM Solutions for a small planet.

LAN DISTANCE CONNECTION SERVER



Establishes a connection and routes traffic between the LAN and WAN
Enforces security on/off the LAN
Manages the "outer" network
Provides audit trails and administration
LAN workstation to dial off LAN

IBM Solutions for a small planet.

LAN DISTANCE SUPPORTED CONNECTIVITIES

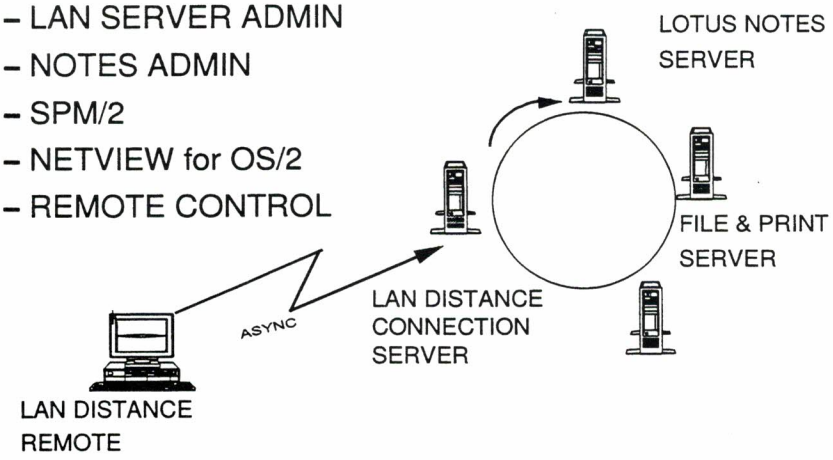
LOCAL AREA NETWORK
- TOKEN RING & ETHERNET

- WIDE AREA NETWORK
  - ASYNCHRONOUS
  - SYNCHRONOUS
  - ISDN
  - X.25
  - CBX/PBX (ROLM)
- NDIS INTERFACE (LAN/WAN)

IBM Solutions for a small planet.

REMOTE ADMINISTRATION

- NETWARE ADMIN
- LAN SERVER ADMIN
- NOTES ADMIN
- SPM/2
- NETVIEW for OS/2
- REMOTE CONTROL



IBM Solutions for a small planet.

Other Tools

LMU

- Remote program execution
- Remote shut-down
- Scheduler

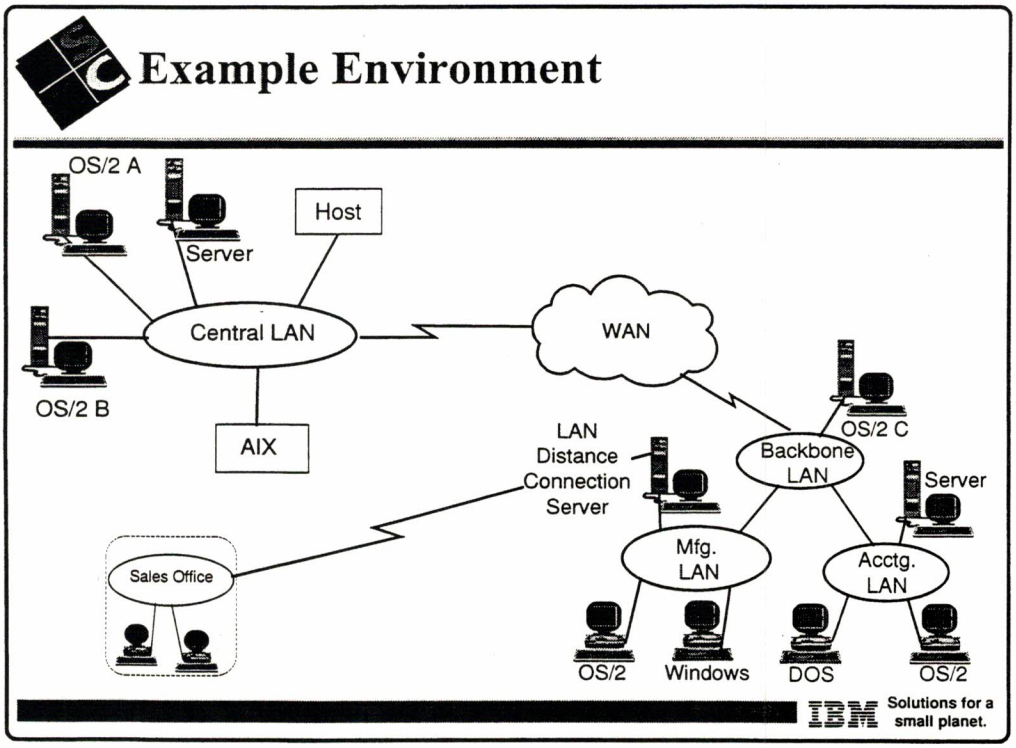
• NetView for OS/2

- Remote command line
- MIB browser

• NetFinity

- Remote screen capture
- Remote session

IBM Solutions for a small planet.


Example Environment
OS/2 A
Server
Host
Central LAN
OS/2 B
AIX
WAN
OS/2 C
LAN
Distance
Connection
Server
Backbone
LAN
Server
Mfg.
LAN
Acctg.
LAN
Sales Office
OS/2
Windows
DOS
OS/2
IBM
Solutions for a
small planet.